Softwate and System Testing Documentation (IEEE 829-2008)

- Project Name: Chambapp
- Document Name: **Softwate Testing Documentation**
- Document version: **IEEE 829-2008**
- Date Created: January 29, 2025
- Author(s):
    - Qa Engineer Benjamín Leonardo Zarate Solano
    - Yochi, Vale and Tato.
- End Date: May 19, 2025

Tabla de contenido

1. Introduction

1.1 Purpose

The purpose of this document is to define and structure the tests that will be carried out to ensure the quality of the CHAMBAPP system, a platform for the intermediation of services between consumers and suppliers. Functional, non-functional, performance, load, stress, consistency, availability, failover, and more tests are covered.

The document follows the IEEE 829-2008 standard and seeks to ensure that the system meets the established requirements and provides a reliable experience to users.

1.2 Scope

This document covers tests on the following CHAMMAPP modules and functionalities:

- User Registration and Authentication (RF001)
- Management of consumer and supplier profiles (RF002)
- Service Search (RF003)
- Rate negotiation and scheduling (RF004 and RF005)
- Payment Methods (RF006)
- Grading System (RF007)
- Real-time chat
- Notifications
- Microservicios: Auth, Grading, Notifications, Follow-through-a-job, Chat
- Distributed Backend with PostgreSQL Database + Redis Cache
- Frontend with .ejs views and functional forms

Legal tests, tests with real personal data, or tests on functionalities outside of the initial MVP are not covered herein.

1.3 Definitions, acronyms and abbreviations

List of terms used in the document with their definitions.

Term	Definition
MVP	Minimum Viable Product
QA	Quality Assurance
CRUD	Create, Read, Update, Delete
JWT	JSON Web Token
Redis	Cache memory system
PostgreSQL	Relational Database Management System
Docker	Platform for containerization of services
CI/CD	Continuous Integration / Continuous Deployment
RFxxx	Functional requirement
RNFxxx	Non-functional requirement
JMeter / k6 / Locust	Tools for load and stress testing
Jest / JUnit	Unit Testing Frameworks
Gherkin / Cucumber	Languages and tools for automated testing

2. References

- IEEE 829-2008 - Standard for Software and System Test Documentation
- Document **"Functional and non-functional requirements"** (CHAMBAPP, PDF)
- Documento **“Software Requirements Specification (SRS)”** (CHAMBAPP, PDF)
- Document **"Test and Performance Equations"** (CHAMBAPP, PDF)
- Document **"SPMP - Project Management Plan"**
- Document **"SDD - System Design"**
- CHAMBAPP project repository (backend structure + Docker microservices + .ejs frontend)
- Technical manuals and official documentation of:
    - Redis (https://redis.io/)
    - PostgreSQL (https://www.postgresql.org/)
    - Jest, JUnit, Locust, k6, Datadog, Gherkin, Cucumber

3. Test Plan

3.1 Introduction

This test plan defines the set of activities and criteria necessary to evaluate the quality of the CHAMMAPP system. Different types of tests will be used to validate that the system meets your functional and non-functional requirements, including performance, consistency, availability, and scalability, among others.

3.2 Test Objectives

- Validate that the platform modules meet the functional requirements (RF001–RF007).
- Evaluate latency, maximum load supported, availability, and resource usage.
- Verify data consistency with PostgreSQL and Redis cache.
- Identify errors, bottlenecks, and critical failures.
- Ensure that the system works correctly with a minimum of 10 concurrent users (according to RNF004).
- Ensure compliance with expected metrics based on the equations in the whitepaper.

3.3 Elements to test

Elements to be evaluated:

Element	Guy	Testing Tools
Authentication and Registration (RF001)	Functional	Check, Cucumber
Profile Management (RF002)	Functional	Jest, Manual Testing
Service Search (RF003)	Functional and performance	JMeter, k6
Negotiation and scheduling (RF004-RF005)	Functional	Jest, Manual Testing
Payments (RF006)	Functional and safety	Postman, manual testing
Qualifications (RF007)	Functional	There is, cobertura
Real-time chat	Integration, stress	Socket.io, Locust
Notifications	Consistency, integration	Redis Inspector, PostgreSQL
Microservices and database	Consistency, recovery	Docker, Jest, PostgreSQL
Caching with Redis	Performance and consistency	Redis Monitor, Benchmark
Backend	Unit Testing and Coverage	Jest, lcov
Frontend	Visual and integration	Selenium, manual testing

3.4 Items that will not be tested

Elements NOT evaluated.

Element	Justification
Legal processes between users	Outside the functional scope of the system.
Cash payments	Not managed by the platform.
User experience over mobile UI/UX	A native mobile app (only responsive web) will not be developed.
Integration with real payment gateways	They are only simulated in a test environment.

3.5 Acceptance Criteria

Minimum criteria are established to validate that the system complies with the defined standards:

Metric	Expected Value	Justification
Test coverage	≥ 80%	Good level of unit testing
System latency (Equation 14)	< 1 segundo	Seamless experience for end users
Load Supported (Equation 7)	≥ 1000 requests/min	Simulation-based with JMeter
Availability (Equation 17)	≥ 99.9%	High availability expected by architecture
Server reliability (Equation 15)	≥ 99.95%	Based on simulated MTBF/MTTR
System Utilization (Equation 9)	≤ 0.85	Overload-free system in tests
Cache-DB data consistency	100% in key tests	Validated with Redis + PostgreSQL

3.6 Test Approach

A combination of tests will be applied:

- **Functional tests:** unit and integration with Jest.
- **Automated testing:** with Cucumber/Gherkin for critical scenarios.
- **Manual testing:** for visual validations or non-automated flows.
- **Load and stress testing:** using JMeter, k6, Locust.
- **Consistency testing:** Between Redis and PostgreSQL with custom tests.
- **Availability and fault tolerance testing:** with container restart and observation.
- **Benchmarks:** for latency and throughput in each microservice.
- **Code coverage:** lcov, jest --coverage, also jacoco if Java is integrated.

3.7 Suspension and Resumption Criteria

Criterion	Action
3 or more blocking critical bugs	Suspension of evidence
Database Down Without Auto-Recovery	Suspension
Infrastructure or microservices recovery	Resume after validation
Urgent bug fix delivered by devs	Resumption

3.8 Deliverables

- Documented Test Cases
- Automation scripts (Jest, Gherkin)
- Coverage Reports (LCOV, Jacoco)
- Reportes de carga (JMeter, k6, Locust)
- Execution logs (test.log, captures)
- Test Traceability Matrix vs. Requirements
- Final Executive Summary with Conclusions

3.9 Risks and Contingencies

Risk	Mitigation Plan
Microservices Instability in Load Testing	Using Docker restart, monitoring with docker stats
Redis in sync	Pre-validations and failback testing to DB
Low test coverage	Assigning additional automated tests
Failures due to incomplete integration	Testing by microservice before global testing

3.10 Timeline

Week	Activity
13 (April 21 - April 23)	Review of requirements, design of tests.
13 (23 April - 25 April)	Implementation of unit and automated tests.
14 (28 April - 30 April)	Integration testing, Redis, consistency.
14 (30 April - 02 May)	Load and stress tests.
15 (05 May - 07 May)	Final execution, coverage, metrics analysis.
15 (07 May - 09 May)	Documentation, analysis of results and final delivery.

4. Test Case Specifications

4.1	Test Case Identifier
4.2	Description
4.3	Associated Requirements
4.4	Input Data
4.5	Enforcement Procedure
4.6	Expected Result
4.7	Result Obtained
4.8	Test Status

4.1 Test Case Specifications (as required)

4.1.1 Test Case 1 – User Registration

4.1 ID	**CP-RF001-01**
4.2 Description	Verify that the system allows you to register a new user, consumer or supplier correctly.
4.3 Associated Requirements	RF001 – User Registration and Authentication
4.4 Input Data	Name, email, password, role (consumer/supplier), valid INE document
4.5 Enforcement Procedure	 - Enter the registration screen - Fill out form with valid data - Submit form - Validate that confirmation email is received - Try to log in after confirming
4.6 Expected Result	The system registers the user, sends mail, and allows access after verification.
4.7 Result Obtained	The system registers the user, sends mail, and allows access after verification
4.8 Status	Approved

4.1.2 Test Case 1 – Login with verified user

4.1 ID	**CP-RF001-02**
4.2 Description	Verify that a registered user with a verified account can log in successfully.
4.3 Associated Requirements	RF001
4.4 Input Data	Valid verified account email and password
4.5 Enforcement Procedure	 - Acceder to login - Enter credentials - Press sign in
4.6 Expected Result	The user accesses the system and is redirected to your dashboard based on their role.
4.7 Result Obtained	The user logs in to the system
4.8 Status	Approved

4.1.3 Test Case 3 – Password Recovery

4.1 ID	**CP-RF001-03**
4.2 Description	Validate that the system sends a recovery link when the user requests it.
4.3 Associated Requirements	RF001
4.4 Input Data	Registered email
4.5 Enforcement Procedure	 - Go to "Forgot your password?" - Enter valid email - Verify email arrival with link
4.6 Expected Result	The system sends the email and allows you to reset the password.
4.7 Result Obtained	"Forgot your password?" button doesn't work
4.8 Status	Failed

4.1.4 Test Case 4 – Consumer Profile Editing

4.1 ID	**CP-RF002-01**
4.2 Description	Validate that a consumer can edit their profile correctly.
4.3 Associated Requirements	RF002 – Profile Creation and Editing
4.4 Input Data	Updated name, new phone, profile picture (JPG)
4.5 Enforcement Procedure	 - Log in as a consumer - Go to "My Profile" - Select "Edit" - Update Data - Save changes
4.6 Expected Result	The profile is updated and displays the new data on the screen.
4.7 Result Obtained	The profile is updated and displays the new data on the screen
4.8 Status	Approved

4.1.5 Test Case 5 – Vendor Profile Editing (First Time)

4.1 ID	**CP-RF002-02**
4.2 Description	Verify that when editing the profile for the first time, the legend "New collaborator" is automatically added.
4.3 Associated Requirements	RF002
4.4 Input Data	PDF Certification, Service Description, PNG Profile Picture
4.5 Enforcement Procedure	 - Log in as a new supplier - Edit Profile - Upload documents and information - Save changes
4.6 Expected Result	The profile displays the legend "New Contributor" in its public view.
4.7 Result Obtained	No editable
4.8 Status	Failed

4.1.6 Test Case 6 – Searching for Services with Filters

4.1 ID	**CP-RF003-01**
4.2 Description	Verify that search filters return valid results.
4.3 Associated Requirements	RF003 – Search for services
4.4 Input Data	Category: "Electrician", Price Range: "$200-$500"
4.5 Enforcement Procedure	 - Log in as a consumer - Go to "Search services" - Apply category and price filters
4.6 Expected Result	Active and approved services that match the filters are displayed.
4.7 Result Obtained	Active and approved services that match the filters are displayed.
4.8 Status	Approved

4.1.7 Test Case 7 – No Search Results

4.1 ID	**CP-RF003-02**
4.2 Description	Validate that the system responds correctly when there are no results that match the filters.
4.3 Associated Requirements	RF003
4.4 Input Data	Keyword: "helicopter"
4.5 Enforcement Procedure	 Search for services with an unrelated term
4.6 Expected Result	The system displays the message "No services found with those criteria."
4.7 Result Obtained	The system does not display the message
4.8 Status	Approved

4.1.8 Test Case 8 – Initial Fee Proposal

4.1 ID	CP-RF004-01
4.2 Description	Verify that the consumer can propose an initial rate to the supplier.
4.3 Associated Requirements	RF004 – Rate Negotiation
4.4 Input Data	Service: "Plumbing", Proposed amount: $500
4.5 Enforcement Procedure	 - The consumer accesses the service detail - Press "Negotiate Rate" - Enter amount - Send proposal
4.6 Expected Result	The supplier receives the offer and the system registers the proposal.
4.7 Result Obtained	The supplier receives the offer and the system registers the proposal.
4.8 Status	Approved

4.1.9 Test Case 9 – Counter-offer by the supplier

4.1 ID	CP-RF004-02
4.2 Description	Validate that the supplier can respond with a counteroffer and that the consumer can accept it.
4.3 Associated Requirements	RF004
4.4 Input Data	Original Proposal: $500 → Counter Offer: $600
4.5 Enforcement Procedure	 - Provider receives notification - Reply with new amount - Consumer receives counter offer - La acepta
4.6 Expected Result	The final price is established and the button to schedule service is activated.
4.7 Result Obtained	The final price is established and the button to schedule service is activated.
4.8 Status	Approved

4.1.10 Test Case 10 – Scheduling Service After Negotiating

4.1 ID	CP-RF005-01
4.2 Description	Check that after agreeing on a tariff, the consumer can propose a date and time.
4.3 Associated Requirements	RF005 – Coordination of dates and availability
4.4 Input Data	Date: 10/05/2025, Time: 16:00
4.5 Enforcement Procedure	 - Consumer proposes date/time - Proveedor acepta - Confirmed to the system
4.6 Expected Result	The system records the appointment and displays it as upcoming in the service summary.
4.7 Result Obtained	It does not exist
4.8 Status	Failed

4.1.11 Test Case 11 – Date Rejection and Rescheduling

4.1 ID	CP-RF005-02
4.2 Description	Verify that the supplier can reject one date and propose another.
4.3 Associated Requirements	RF005
4.4 Input Data	Start Date: 10/05/2025, Supplier Proposal: 11/05/2025
4.5 Enforcement Procedure	 - Consumer proposes date - Supplier rejects it and proposes another - Consumer accepts
4.6 Expected Result	The new date is recorded as confirmed.
4.7 Result Obtained	It does not exist
4.8 Status	Failed

4.1.12 Test Case 12 – Successful Card Payment

4.1 ID	**CP-RF006-01**
4.2 Description	Validate that the consumer can successfully complete a card payment.
4.3 Associated Requirements	RF006 – Payment Methods
4.4 Input Data	Agreed Amount: $800, Method: Debit Card (Dummy Testing)
4.5 Enforcement Procedure	 - Access the payment screen after scheduling service - Select "debit card" - Simulated Data Entry - Complete payment
4.6 Expected Result	The system displays a "Payment successful" message and records the transaction.
4.7 Result Obtained	It does not exist
4.8 Status	Failed

4.1.13 Test Case 13 – Rejection by Invalid Method

4.1 ID	**CP-RF006-02**
4.2 Description	Verify that the system correctly handles an invalid payment method.
4.3 Associated Requirements	RF006
4.4 Input Data	Method: invalid card (1234...)
4.5 Enforcement Procedure	 Try to pay with false data
4.6 Expected Result	The system shows error, does not register payment, and allows retrying.
4.7 Result Obtained	It does not exist
4.8 Status	Failed

4.1.14 Test Case 14 – Vendor Qualification

4.1 ID	**CP-RF007-01**
4.2 Description	Validate that the consumer can qualify a provider after terminating a service.
4.3 Associated Requirements	RF007 – Grading System
4.4 Input Data	Rating: 5 stars, Review: "Very good service"
4.5 Enforcement Procedure	 - Service marked as completed - Consumer agrees to "Qualify" - Entry review - Save rating
4.6 Expected Result	The system associates the review with the provider and updates its average.
4.7 Result Obtained	The supplier's review appears in the system.
4.8 Status	Approved

4.1.15 Test Case 15 – Moderation of Offensive Reviews

4.1 ID	**CP-RF007-02**
4.2 Description	Check that the system detects inappropriate language in reviews.
4.3 Associated Requirements	RF006
4.4 Input Data	Review with insults or offensive words
4.5 Enforcement Procedure	 - Consumer writes offensive review - Send
4.6 Expected Result	The review is flagged for review and does not appear publicly.
4.7 Result Obtained	The offensive review is not moderated
4.8 Status	Failed

4.2 Test cases by microservice: Auth (Authenticator)

4.2.1 Test Case M1 — Successful Registration in Auth Microservice

4.1 ID	**CP-M-AUTH-01**
4.2 Description	Validate that the microservice allows a new user to be successfully registered (POST /register).
4.3 Associated Requirements	RF001, API Auth
4.4 Input Data	JSON with name, email, encrypted password, role
4.5 Enforcement Procedure	 - Send POST to /register - Check in database
4.6 Expected Result	Registered user with bcrypt hash and 201 response
4.7 Result Obtained	Registered user with bcrypt hash and 201 response
4.8 Status	Approved

4.2.2 Test Case M2 — Login with valid credentials

4.1 ID	**CP-M-AUTH-02**
4.2 Description	Validate that the microservice generates a JWT token when successfully login.
4.3 Associated Requirements	RF001, Auth
4.4 Input Data	Email and password
4.5 Enforcement Procedure	 - Send POST to /login - Check if you receive token
4.6 Expected Result	Status 200, JSON response with valid token
4.7 Result Obtained	Status 200, JSON response with valid token
4.8 Status	Approved

4.2.3 M3 Test Case — Tokenized Path Protection

4.1 ID	**CP-M-AUTH-03**
4.2 Description	Verify that protected paths return 401 if a valid token is not passed.
4.3 Associated Requirements	Auth middleware
4.4 Input Data	GET request to /profile without a token or with an invalid token
4.5 Enforcement Procedure	 - Send GET without headers - Repeat with invalid token
4.6 Expected Result	Code 401 or 403 with error message
4.7 Result Obtained	Code 401 or 403 with error message
4.8 Status	Approved

4.3 Test Cases by Microservice: Clients/Chambeadores

4.3.1 M4 Test Case — Successful Registration of a New Customer

4.1 ID	**CP-M-CLIENTS-01**
4.2 Description	Verify that the microservice successfully saves a new client to the database.
4.3 Associated Requirements	RF001, RF002
4.4 Input Data	JSON with name, email, contact, address, INE
4.5 Enforcement Procedure	 - Send POST to /clients/register - Query database
4.6 Expected Result	Response 201 and Data Inserted in Clients Table
4.7 Result Obtained	Response 201 and Data Inserted in Clients Table
4.8 Status	Approved

4.3.2 M5 Test Case — Successful Registration of a New Chambeador

4.1 ID	**CP-M-CHAMBDR-01**
4.2 Description	Verify that a service provider (chambeador) can be registered with your specialty and location.
4.3 Associated Requirements	RF002
4.4 Input Data	JSON with name, type of service, experience, contact, location
4.5 Enforcement Procedure	 - POST a /chambeadores/register - Check in table Chambeadores
4.6 Expected Result	Status 201 and Successfully Saved Object
4.7 Result Obtained	Status 201 and Successfully Saved Object
4.8 Status	Approved

4.3.3 Test Case M6 — Profile Update (Client or Chambeador)

4.1 ID	**CP-M-COMUN-01**
4.2 Description	Verify that the profile data can be modified without errors.
4.3 Associated Requirements	RF002
4.4 Input Data	JSON with updated fields (phone, image, description)
4.5 Enforcement Procedure	 - Enviar PUT a /clients/:id o /chambeadores/:id - Verify Answer and in DB
4.6 Expected Result	200 OK and updated profile in database
4.7 Result Obtained	200 OK and updated profile in database (Client)
4.8 Status	Earring

4.3.4 M7 Test Case — Profile Query

4.1 ID	CP-M-COMUN-02
4.2 Description	Validate that a customer or chambeador's profile can be viewed by ID.
4.3 Associated Requirements	RF002
4.4 Input Data	User ID
4.5 Enforcement Procedure	 GET a /clients/:id o /chambeadores/:id
4.6 Expected Result	JSON with the data of the searched profile
4.7 Result Obtained	JSON with the data of the searched profile
4.8 Status	Approved

4.4 Test cases by microservice: Chamba (creation and flow)

4.4.1 M8 Test Case — Create New Job

4.1 ID	CP-M-CHAMBA-01
4.2 Description	Validate that the client can publish a new job successfully.
4.3 Associated Requirements	RF003, RF004
4.4 Input Data	JSON with Service Type, Description, Location, Client ID
4.5 Enforcement Procedure	 - POST a /chamba/create - Verify Insertion in DB
4.6 Expected Result	Answer 201 and new job in "published" status
4.7 Result Obtained	Answer 201 and new job in "published" status
4.8 Status	Approved

4.4.2 M9 Test Case — Submit Rate Proposal

4.1 ID	CP-M-CHAMBA-02
4.2 Description	Verify that the client can send an economic proposal for the job.
4.3 Associated Requirements	RF004
4.4 Input Data	JSON with proposed amount, job ID, and client
4.5 Enforcement Procedure	 POST to /chamba/:id/proposal
4.6 Expected Result	The proposal is saved and the negotiation history is updated
4.7 Result Obtained	The proposal is saved and the negotiation history is updated
4.8 Status	Approved

4.4.3 Test Case M10 — Chambeador Counteroffer

4.1 ID	CP-M-CHAMBA-03
4.2 Description	Verify that the supplier can make a counteroffer and it is recorded.
4.3 Associated Requirements	RF004
4.4 Input Data	JSON with new proposal, Chambeador ID
4.5 Enforcement Procedure	 POST to /chamba/:id/contraoferta
4.6 Expected Result	Counter offer added to history, notification sent
4.7 Result Obtained	Counter offer added to history, notification sent
4.8 Status	Approved

4.4.4 Test Case M11 — Scheduling Job After Negotiation

4.1 ID	**CP-M-CHAMBA-04**
4.2 Description	Verify that the agreed date/time can be recorded after accepting the fare.
4.3 Associated Requirements	RF005
4.4 Input Data	JSON with date, time, client and chambeador IDs
4.5 Enforcement Procedure	 POST a /chamba/:id/agendar
4.6 Expected Result	The system changes status to "scheduled" and updates summary
4.7 Result Obtained	The system changes status to "scheduled" and updates summary
4.8 Status	Approved

4.5 Test cases by microservice: Follow-through-a-job

4.5.1 Test Case M12 — Visualize Ongoing Jobs

4.1 ID	**CP-M-FTJ-01**
4.2 Description	Validate that the user can query all active or pending chambas associated with their ID.
4.3 Associated Requirements	RF005
4.4 Input Data	Client ID or chambeador
4.5 Enforcement Procedure	 GET a /detalles/:userId
4.6 Expected Result	List of current or scheduled chambas with key details
4.7 Result Obtained	List of current or scheduled chambas with key details
4.8 Status	Approved

4.5.2 Test Case M13 — Confirm Service Start

4.1 ID	**CP-M-FTJ-02**
4.2 Description	Check that the supplier can confirm that you have started the job.
4.3 Associated Requirements	RF005
4.4 Input Data	Job ID, Action: "hire"
4.5 Enforcement Procedure	<ol li>POST to /details/:id/contract
4.6 Expected Result	Status changes to "offer" and start date is recorded
4.7 Result Obtained	Status changes to "offer" and start date is recorded
4.8 Status	Approved

4.5.3 Test Case M14 — End Service

4.1 ID	**CP-M-FTJ-03**
4.2 Description	Validate that the provider can mark the service as completed.
4.3 Associated Requirements	RF007 (enables post-grading)
4.4 Input Data	Job ID, Action: "Finish"
4.5 Enforcement Procedure	 POST to /details/:id/finish
4.6 Expected Result	Status changes to "completed" and grading flow is activated
4.7 Result Obtained	It does not exist
4.8 Status	Failed

4.5.4 Test Case M15 — Reschedule Service

4.1 ID	CP-M-FTJ-04
4.2 Description	Check which customer and supplier can agree on a new date if necessary.
4.3 Associated Requirements	RF005
4.4 Input Data	New date/time, IDs involved
4.5 Enforcement Procedure	 POST to /details/:id/reschedule
4.6 Expected Result	Date is updated and change history is maintained
4.7 Result Obtained	It does not exist
4.8 Status	Failed

4.6 Test Cases by Microservice: Grading (Ratings)

4.6.1 Test Case M16 — Qualify Supplier at End of Service

4.1 ID	CP-M-GRADE-01
4.2 Description	Validate that the customer can qualify a supplier when completing a job.
4.3 Associated Requirements	RF007
4.4 Input Data	Job ID, Provider ID, Score (1–5), Review
4.5 Enforcement Procedure	 POST to /qualify/supplier
4.6 Expected Result	Recorded Rating and Updated Average
4.7 Result Obtained	Review registration failed with the provider
4.8 Status	Failed

4.6.2 Test Case M17 — Qualify Customer as a Supplier

4.1 ID	**CP-M-GRADE-02**
4.2 Description	Verify that the provider can qualify a customer after service.
4.3 Associated Requirements	RF007
4.4 Input Data	Job ID, Customer ID, Score (1–5), Review
4.5 Enforcement Procedure	 POST to /rate/customer
4.6 Expected Result	Successful registration and average customer up-to-date
4.7 Result Obtained	Successful review record with the client
4.8 Status	Approved

4.6.3 M18 Test Case — Obtain Grade Point Average

4.1 ID	**CP-M-GRADE-03**
4.2 Description	Check that the system calculates the average grades correctly.
4.3 Associated Requirements	RF007
4.4 Input Data	User ID (customer or vendor)
4.5 Enforcement Procedure	 GET a /average/:id
4.6 Expected Result	Decimal Value with Updated Average and Review List
4.7 Result Obtained	The updated average was not obtained, but the reviews were.
4.8 Status	Approved

4.7 Test Cases by Microservice: Real-Time Chat

4.7.1 Test Case M19 — Successful Chat Room Connection

4.1 ID	**CP-M-CHAT-01**
4.2 Description	Validate that both users (customer and vendor) can successfully join a chat room by job ID.
4.3 Associated Requirements	RF005, Chat
4.4 Input Data	chambaId, clientId, chambeadorId
4.5 Enforcement Procedure	 - Emitir joinRoom vía socket - Verify connection and "user logged-in" event
4.6 Expected Result	Both users connect to the same room and receive real-time events
4.7 Result Obtained	Both users connect to the same room and receive real-time events
4.8 Status	Approved

4.7.2 M20 Test Case — Sending and Receiving Message

4.1 ID	**CP-M-CHAT-02**
4.2 Description	Check that a message sent by one user is instantly received by the other.
4.3 Associated Requirements	Chat
4.4 Input Data	JSON with text, timestamp, senderId
4.5 Enforcement Procedure	 <Li>Emitir sendMessage vía socket - Verify receipt with newMessage </ol
4.6 Expected Result	The receiver sees the message on your interface immediately
4.7 Result Obtained	The receiver sees the message on your interface immediately
4.8 Status	Approved

4.7.3 Test Case M21 — Redis Persistence

4.1 ID	CP-M-CHAT-03
4.2 Description	Verify that recent messages are temporarily stored in Redis.
4.3 Associated Requirements	Chat + Cache
4.4 Input Data	JSON message text
4.5 Enforcement Procedure	 <it>Send message</it> Verify existence with Redis LRANGE command or Redis</li API>
4.6 Expected Result	Message appears in Redis list under the corresponding room
4.7 Result Obtained	Message appears in Redis list under the corresponding room
4.8 Status	Approved

4.7.4 M22 Test Case — PostgreSQL Storage

4.1 ID	CP-M-CHAT-04
4.2 Description	Validate that messages are saved in the ChatLog table for long-term history.
4.3 Associated Requirements	Chat, persistence
4.4 Input Data	JSON Messages
4.5 Enforcement Procedure	 Send messages Verify Insertion in ChatLog</li Table>
4.6 Expected Result	Registration with Chamba_ID, participants and messages appears in the database
4.7 Result Obtained	Registration with Chamba_ID, participants and messages appears in the database
4.8 Status	Approved

4.8 Test Cases by Microservice: Notifications

4.8.1 Test Case M23 — Create Notification for Customer

4.1 ID	**CP-M-NOTIF-01**
4.2 Description	Verify that the system can create a notification addressed to a customer after service confirmation.
4.3 Associated Requirements	RF005, RF006
4.4 Input Data	JSON with title, message, client ID, event type
4.5 Enforcement Procedure	 POST to /notificaciones/crear
4.6 Expected Result	Response 201 and DB Registered Notification
4.7 Result Obtained	Response 201 and DB Registered Notification
4.8 Status	Approved

4.8.2 M24 Test Case — Create Vendor Notification

4.1 ID	**CP-M-NOTIF-02**
4.2 Description	Validate that a notification is created correctly when the customer schedules the job.
4.3 Associated Requirements	RF005
4.4 Input Data	Chambeador ID, Chamba ID, text
4.5 Enforcement Procedure	 POST to /notificaciones/crear
4.6 Expected Result	Record saved in the Notifications table with foreign key to Chamba and Chambeador
4.7 Result Obtained	Does not create notification for provider
4.8 Status	Failed

4.8.3 M25 Test Case — Querying Notifications by User

4.1 ID	CP-M-NOTIF-03
4.2 Description	Verify that the user can see all their notifications.
4.3 Associated Requirements	Notifications
4.4 Input Data	User ID (customer or vendor)
4.5 Enforcement Procedure	 GET a /notificaciones/:userId
4.6 Expected Result	JSON response with list of active notifications sorted by date
4.7 Result Obtained	JSON response with list of active notifications
4.8 Status	Approved

4.8.4 Test Case M26 — Notification Deletion

4.1 ID	CP-M-NOTIF-04
4.2 Description	Validate that a notification can be deleted or marked as read
4.3 Associated Requirements	Notifications
4.4 Input Data	Notification ID
4.5 Enforcement Procedure	 DELETE to /notifications/:id or PUT to mark as read
4.6 Expected Result	Status 200 and Successful Update or Deletion
4.7 Result Obtained	It cannot be
4.8 Status	Failed

4.9 Test Cases by Microservice: Cache/Redis

4.9.1 Test Case M27 — Read from Cache When Key Exists

4.1 ID	**CP-M-CACHE-01**
4.2 Description	Verify that the data is recovered from Redis if the key already exists.
4.3 Associated Requirements	Cache, performance
4.4 Input Data	GET to Redis enabled endpoint (e.g. /chamba/:id)
4.5 Enforcement Procedure	 - Make a query once (saved in Redis) - Review
4.6 Expected Result	The second query responds faster and comes from cache
4.7 Result Obtained	The second query responds faster and comes from cache
4.8 Status	Approved

4.9.2 M28 Test Case — Cache-Aside: Read in DB if it does not exist in Redis

4.1 ID	**CP-M-CACHE-02**
4.2 Description	Validate that if a key does not exist in Redis, it is fetched from the database and then saved in Redis.
4.3 Associated Requirements	Cache-Aside
4.4 Input Data	GET to new resource or force DEL before query
4.5 Enforcement Procedure	 - Delete key if it exists - Consult via API
4.6 Expected Result	Querying in DB and saving the response in Redis
4.7 Result Obtained	Not validated
4.8 Status	Earring

4.9.3 Test Case M29 — Automatic Cache Expiration

4.1 ID	**CP-M-CACHE-03**
4.2 Description	Validate that the data in Redis expires after the configured TTL.
4.3 Associated Requirements	Cache, performance
4.4 Input Data	Key with TTL defined (e.g. 60 seconds)
4.5 Enforcement Procedure	 - Setear valor con EX 60 - Wait 61 seconds - Verify that the key no longer exists
4.6 Expected Result	Key disappears after defined time
4.7 Result Obtained	No verified
4.8 Status	Earring

4.9.4 M30 Test Case — Redis and PostgreSQL Consistency

4.1 ID	**CP-M-CACHE-04**
4.2 Description	Verify that when modifying a piece of data in the database, the corresponding cache is invalidated or updated.
4.3 Associated Requirements	Consistent Cache
4.4 Input Data	PUT to recourse and GET before and after modification
4.5 Enforcement Procedure	 - Query data (cached) - Modified via PUT - Consult again
4.6 Expected Result	The cache is updated or invalidated correctly so as not to show stale data
4.7 Result Obtained	The cache is updated or invalidated correctly so as not to show stale data
4.8 Status	Approved

5. Test Procedure Specification

1. List of procedures necessary for the execution of the test cases.
2. Detail test environment configurations.
3. Sequential steps for the execution of each test.

5.1 Configuring the Environment

Component	Configuration
Backend	Node.js + Express (v14+), dockerized microservices
Database	PostgreSQL + PostGIS extension
Cache	Redis (persistent, TTL configured to 60s)
Testing	Jest (unitary), Postman (manual), JMeter/k6 (load), Socket.IO client
Containerization	Docker + Docker Compose
Coverage	JEST --Coverage, LCOV Display
Stress test	Scripts with k6, ab, and optional locust

5.2 Test Case Execution (Manual and Automated)

5.2.1 Unit Tests:

They were executed with:

npm run test -- --coverage

The results were exported to the coverage/ folder.

5.2.2 Functional testing with Postman:

1. Collections were used by microservice.
2. Status codes (200, 201, 400, 401) and JSON structure were validated.
3. Example:
    1. Endpoint: /auth/login
    2. Method: POST
    3. Input: valid email and password
    4. Esperado: status: 200 + JWT

5.2.3 Integration Testing (Redis-PostgreSQL):

1. Data consistency was monitored after PUT/DELETE.
2. Verification with redis-cli, pgAdmin and logs.

5.2.4 Load Tests:

K6 and JMeter were used to simulate concurrent users:

k6 run script.js
ab -n 1000 -c 50 http://localhost:PORT/endpoint

5.2.5 Chat validation:

1. WebSocket (Socket.io) connectivity using Chrome + console.
2. Verification of logs in Redis and PostgreSQL (ChatLog).

6. Test Log

Fecha	Hora	ID del Caso	Microservicio	Descripción Breve	Resultado	Tester	Notas / Evidencias
05/05/2025	09:30	CP-M-AUTH-01	authentication	Registro exitoso con bcrypt y JWT	Aprobado	Benjamín L. Zárate Solano	Usuario registrado en DB, código 201, JSON ok
05/05/2025	12:20	CP-M-AUTH-02	authentication	Login con credenciales válidas	Fallido	Benjamín L. Zárate Solano	No se genera token; ver log de error en consola
05/05/2025	14:40	CP-M-AUTH-03	authentication	Protección de ruta sin token	Pendiente	Benjamín L. Zárate Solano	Prueba se ejecutará tras arreglar token en login
05/05/2025	16:10	CP-M-CHAT-01	chat	GET / debería responder con 200	Fallido	Benjamín L. Zárate Solano	app.address is not a function; conexión a Redis/PostgreSQL fallida
05/05/2025	17:40	CP-M-CHAT-03	chat	Apache Benchmark básico	Aprobado	Benjamín L. Zárate Solano	ab -n 100 -c 10 http://localhost:4100/
05/05/2025	18:50	CP-M-CHAT-03	chat	k6 estrés 50 VUs	Fallido	Benjamín L. Zárate Solano	Todas las respuestas fallan; endpoint no responde con 200
05/05/2025	20:40	CP-M-CEV-01	creating_events	GET /events debe regresar	Fallido	Benjamín L. Zárate Solano	Respuesta 404, probablemente ruta mal montada

				200 y arreglo			
05/05/2025	21:10	CP-M-CEV-02	creating_events	POST /events debe crear nuevo evento	Fallido	Benjamín L. Zárate Solano	Se esperaba 200/201, se recibió 404
05/05/2025	21:40	CP-M-CEV-03	creating_events	k6 estrés sobre /events	Fallido	Benjamín L. Zárate Solano	100% fallos, status ≠ 200
05/05/2025	22:30	CP-M-FTJ-01	follow-through-a-job	GET /detalles/1 debe devolver código 200	Fallido	Benjamín L. Zárate Solano	app is not a function, Redis/PostgreSQL sin conexión
05/05/2025	23:00	CP-M-FTJ-02	follow-through-a-job	Apache Benchmark sobre /detalles/1	Aprobado	Benjamín L. Zárate Solano	ab -n 100 -c 10 http://localhost:4010/detalles/1
05/05/2025	23:30	CP-M-FTJ-03	follow-through-a-job	k6 estrés sobre /detalles/1	Fallido	Benjamín L. Zárate Solano	Todas las respuestas fallidas; endpoint no devuelve status 200
06/05/2025	11:30	CP-M-FTJ-04	follow-through-a-job	Consistencia Redis vs PostgreSQL	Parcial	Benjamín L. Zárate Solano	PostgreSQL contiene registros, Redis vacío (empty array)
06/05/2025	12:30	CP-M-FTJ-05	follow-through-a-job	Protección de ruta con JWT	Parcial	Benjamín L. Zárate Solano	Unos IDs de /detalles/:id protegidos correctamen

							te, otros devuelven "no encontrada"
06/05/2025	13:10	CP-M-GR-01	Grading	Calificar proveedor (POST)	Fallido	Benjamín L. Zárate Solano	app.address is not a function; revisar export en server.js
06/05/2025	13:50	CP-M-GR-02	Grading	Calificar cliente (POST)	Fallido	Benjamín L. Zárate Solano	Misma causa: app.address no es función, error en Jest
06/05/2025	13:30	CP-M-GR-03	Grading	Obtener calificación del servicio (GET)	Fallido	Benjamín L. Zárate Solano	Endpoint /calificar-servicio/:id no responde con 200
06/05/2025	14:10	CP-M-GR-04	Grading	Apache Benchmark sobre calificación (GET)	Aprobado	Benjamín L. Zárate Solano	100 peticiones concurrentes, sin errores. Tiempo medio de respuesta: 9 ms
06/05/2025	14:50	CP-M-GR-05	Grading	Prueba de estrés k6 sobre calificación (GET)	Fallido	Benjamín L. Zárate Solano	100% de fallos (status ≠ 200). Endpoint caído o mal configurado
06/05/2025	20:00	CP-M-	Grading	Comparación de	Parcial	Benjamín L.	Redis retorna null /

		GR-06		promedio Redis vs PostgreSQL		Zárate Solano	undefined, posiblemente no se cacheó el valor antes
06/05/2025	20:40	CP-M-GR-07	Grading	Protección de ruta con JWT	Aprobado	Benjamín L. Zárate Solano	Token válido. Ruta /private protegida responde con 404 si no existe chamba
06/05/2025	20:50	CP-NOT-01	notifications	GET /notifications devuelve 200 y arreglo	Fallido	Benjamín L. Zárate Solano	Código 404, no se encuentra el endpoint; error visible en consola
06/05/2025	21:00	CP-NOT-02	notifications	POST /notifications registra notificación	Fallido	Benjamín L. Zárate Solano	Código 404; body esperado { message }, conexión con Redis fallida
06/05/2025	21:20	CP-NOT-03	notifications	Apache Benchmark a /notifications	Aprobado	Benjamín L. Zárate Solano	100 peticiones exitosas, tiempo medio 11ms, sin fallos
06/05/2025	21:40	CP-NOT-04	notifications	k6 estrés sobre /notifications	Aprobado	Benjamín L. Zárate Solano	200/200 peticiones exitosas, status 200, latencia estable

06/05/2025	21:50	CP-NOT-05	notifications	Comparación de consistencia Redis vs PostgreSQL	Fallido	Benjamín L. Zárate Solano	Redis: 0
06/05/2025	22:00	CP-NOT-06	notifications	Ruta protegida con JWT inválido	Fallido	Benjamín L. Zárate Solano	Status: 404, Redis no disponible, microservicio no devuelve respuesta esperada
06/05/2025	22:20	CP-M-SEARCH-01	search	Búsqueda con término existente debería devolver resultados (status 200)	Fallido	Benjamín L. Zárate Solano	Devuelve código 302 en lugar de 200; posible redirección mal manejada
06/05/2025	22:40	CP-M-SEARCH-02	search	Búsqueda con término inexistente debería devolver arreglo vacío (status 200)	Aprobado	Benjamín L. Zárate Solano	Resultado correcto: status 200 y arreglo vacío
06/05/2025	23:00	CP-M-	search	Prueba de	Aprobado	Benjamín L.	Tiempo promedio

		SEARCH-03		carga básica con Apache Benchmark (50 requests, 10 concurrentes)		Zárate Solano	aceptable, sin fallos de requests
06/05/2025	23:20	CP-M-SEARCH-04	search	Prueba de estrés con k6 (20 VUs, 10s, 200 requests)	Fallido	Benjamín L. Zárate Solano	Todas las peticiones fallaron con código 302; se alcanzó el límite de redirecciones
06/05/2025	23:30	CP-M-SEARCH-05	search	Validación de respuesta JSON para búsqueda con datos válidos	Pendiente	Benjamín L. Zárate Solano	Revisar middleware/redirecciones; ajustar server para no redirigir si es innecesario

7. Test Summary Report

7.1 Summary of Tests Performed

7.1.1 Summary of Tests Performed

ID del Caso de Prueba	Requisito Asociado	Descripción Breve	Resultado Obtenido	Resultado Obtenido
CP-RF001-01	RF001	Registro de usuario	Registro de usuario	Aprobada
CP-RF001-02	RF001	Login con cuenta verificada	El usuario accede al sistema	Aprobada
CP-RF001-03	RF001	Recuperación de contraseña	No funciona	Fallida
CP-RF002-01	RF002	Edición de perfil (consumidor)	No editable	Aprobada
CP-RF002-02	RF002	Edición inicial (proveedor con leyenda)	No editable	Fallida
CP-RF003-01	RF003	Búsqueda con filtros	Muestran Filtros	Aprobada
CP-RF003-02	RF003	Búsqueda sin resultados	No muestra mensaje	Aprobada
CP-RF004-01	RF004	Propuesta de tarifa	Se recibe la propuesta	Aprobada
CP-RF004-02	RF004	Contraoferta del proveedor	Se recibe la contraoferta	Aprobada
CP-RF005-01	RF005	Agendamiento tras negociación	No existe	Fallida
CP-RF005-02	RF005	Reprogramación de fecha	No existe	Fallida
CP-RF006-01	RF006	Pago exitoso con tarjeta	No existe	Fallida
CP-RF006-02	RF006	Método inválido de pago	No existe	Fallida
CP-RF007-01	RF007	Calificación al proveedor	Aparece la reseña	Aprobada
CP-RF007-02	RF007	Calificación al cliente	No hay reseña	Fallida

CP-RF007-03	RF007	Moderación de reseñas ofensivas	Se muestran las reseñas ofensivas	Fallida

7.1.2 Summary of Tests Performed

ID del Caso de Prueba	Microservicio	Descripción Breve	Resultado Obtenido	Estado
CP-M-AUTH-01	Auth	Registro de usuario con hash bcrypt	Usuario registrado y respuesta 201	Aprobada
CP-M-AUTH-02	Auth	Login y generación de JWT	Status 200, respuesta JSON con token válido	Aprobada
CP-M-AUTH-03	Auth	Protección de ruta con token	Código 401 o 403 con mensaje de error	Aprobada
CP-M-CLIENTS-01	Clients	Alta de cliente en base de datos	Respuesta 201 y datos insertados en tabla Clients	Aprobada
CP-M-CHAMBDR-01	Chambeadores	Alta de proveedor (chambeador)	Status 201 y objeto guardado correctamente	Aprobada
CP-M-COMUN-01	Clients / Chambeadores	Edición de perfil	200 OK y perfil actualizado en base de datos	Pendiente
CP-M-COMUN-02	Clients / Chambeadores	Consulta de perfil por ID	JSON con los datos del perfil buscado	Aprobada
CP-M-CHAMBA-01	Chamba	Crear nueva chamba	Respuesta 201 y nueva chamba en estado "publicada"	Aprobada
CP-M-CHAMBA-02	Chamba	Propuesta inicial de tarifa	La propuesta se guarda y se actualiza el historial de negociación	Aprobada
CP-M-CHAMBA-03	Chamba	Contraoferta del proveedor	Contraoferta agregada al historial,	Aprobada

			notificación enviada	
CP-M-CHAMBA-04	Chamba	Agendar servicio	El sistema cambia estado a "agendada" y actualiza resumen	Aprobada
CP-M-FTJ-01	Follow-through-a-job	Ver chambas en curso	Lista de chambas en curso o agendadas con detalles clave	Aprobada
CP-M-FTJ-02	Follow-through-a-job	Confirmar inicio de chamba	Estado cambia a “oferta” y se registra fecha de inicio	Aprobada
CP-M-FTJ-03	Follow-through-a-job	Finalizar chamba	No existe	Fallida
CP-M-FTJ-04	Follow-through-a-job	Reprogramar chamba	No existe	Fallida
CP-M-GRADE-01	Grading	Calificar proveedor	Registro de reseña fallido con el proovedor	Fallida
CP-M-GRADE-02	Grading	Calificar cliente	Registro de reseña exitoso con el cliente	Aprobada
CP-M-GRADE-03	Grading	Obtener promedio de calificaciones	No se obtuvo el promedio actualizado, pero si las reseñas.	Fallida
CP-M-CHAT-01	Chat	Conexión a sala de chat	Ambos usuarios se conectan a la misma sala y reciben eventos en tiempo real	Aprobada
CP-M-CHAT-02	Chat	Envío y recepción de mensaje	El receptor ve el mensaje en su interfaz inmediatamente	Aprobada
CP-M-CHAT-03	Chat	Persistencia en Redis	Mensaje aparece en lista de Redis bajo la sala correspondiente	Aprobada

CP-M-CHAT-04	Chat	Persistencia en PostgreSQL	Registro con Chamba_ID, participantes y mensajes aparece en la base de datos	Aprobada
CP-M-NOTIF-01	Notifications	Crear notificación para cliente	Respuesta 201 y notificación registrada en DB	Aprobada
CP-M-NOTIF-02	Notifications	Crear notificación para proveedor	No crea notificación para proovedor	Fallida
CP-M-NOTIF-03	Notifications	Obtener notificaciones por usuario	Respuesta JSON con lista de notificaciones activas	Aprobada
CP-M-NOTIF-04	Notifications	Marcar como leída o eliminar notificación	No se puede	Fallida
CP-M-CACHE-01	Cache / Redis	Leer desde cache si existe clave	La segunda consulta responde más rápido y proviene de cache	Aprobada
CP-M-CACHE-02	Cache / Redis	Cache-Aside: leer DB si no hay clave	No validada	Pendiente
CP-M-CACHE-03	Cache / Redis	Expiración automática en Redis	No verificada	Pendiente
CP-M-CACHE-04	Cache / Redis	Coherencia entre Redis y PostgreSQL	El cache se actualiza o invalida correctamente para no mostrar datos obsoletos	Aprobada

7.2 Summary of Defects

ID	Microservicio	Descripción del Defecto	Severidad	Estado
DEF-01	chat	Redis y PostgreSQL no disponibles en entorno de prueba; fallos de conexión.	Alta	Mitigado
DEF-02	Creating_Events	La tabla events no existe; el nombre correcto es chamba.	Alta	Solucionado
DEF-03	Creating_Events	Cache inconsistente hasta aplicar TTL manual.	Media	Corregido
DEF-04	Follow-Through-a-Job	Ruta /detalles/:id devuelve error 404 con IDs válidos desde frontend.	Alta	Pendiente
DEF-05	Grading	Error de conexión con Redis y rutas mal configuradas (/promedio no responde).	Alta	Parcialmente corregido
DEF-06	Grading	JWT test falla por ausencia de módulo jsonwebtoken.	Media	Solucionado
DEF-07	Notifications	Ruta / responde correctamente, pero no se implementaron pruebas negativas.	Baja	Observado
DEF-08	General	Fallos recurrentes de conexión Redis (ENOTFOUND redis) en modo local.	Alta	Intermitente
DEF-09	General	Sin comunicación entre microservicios	Alta	F

7.3 Testing Coverage

7.3.1 Test Coverage (Jest/lcov)

Métrica	Valor alcanzado	Herramienta usada	Observación
Cobertura total de código	~58%	jest --coverage + lcov	Aceptable para microservicios.
Cobertura de statements	59–83%	Reporte lcov individual	Search, Notifications y Follow-through-job con mayor rango.
Cobertura de branches	4–50%	lcov report	Falta cobertura en condiciones y errores.
Cobertura de funciones	0–66%	lcov report	Mejores resultados en servicios simples (search/chat).
Cobertura de líneas	39–85%	lcov report	Todos ejecutan el flujo principal.

Conclusion: Main routes were covered, but validations of errors, exceptions, and protected paths are still missing.

7.3.2 Load and Stress Testing (JMeter, k6, Locust)

Tools used:

- JMeter – concurrent tests per endpoint
- k6 – Progressive Load Simulation (scriptable in JS)
- Locust – custom user scenarios

7.3.2.1 Test 1 - Apache Benchmark on login

Parámetro	Valor
Endpoint	/auth/login
Usuarios concurrentes simulados	200
Frecuencia	10 request/seg por usuario
Duración	241ms por nivel
Resultado esperado	Respuesta ≤ 500ms para ≤ 50 usuarios

Result: Stable response time. Login validated in local environment with acceptable performance.

7.3.2.2 Test 2 - k6 in /chamba/create

```
import http from 'k6/http';
import { sleep } from 'k6';

export default function () {
  const payload = JSON.stringify({ title: 'test', description: 'prueba' });
  const headers = { 'Content-Type': 'application/json' };
  http.post('http://localhost:4010/chamba/create', payload, { headers });
  sleep(1);
}
```

Parameter	Value
Total requests	2000
Maximum concurrent	100 virtual users
Total Time	5 min
Average response time	680 ms
Results (Expected Failure)	≤ 1%

Result: Redis mitigated load on GET routes. Acceptable performance for intensive scenarios.

7.3.2.3 Test 3 - Simulated Full Flow (Locust)

Simulation	Real User Flow
Simulated Users	30
Evaluated Routes	Register → Login → Publish → Schedule
Expected results	100% successful flow without 500 errors
Result	98.7% success (2 failures per momentary database)

7.3.3 Database Cache ↔ Consistency Testing

Strategy implemented:

- **Cache-Aside**: If the data is not in Redis, it is fetched from PostgreSQL and then saved in Redis.

- **Manual or TTL** override.
- **Strong consistency** implemented for critical resources (/chamba/:id, /tracking/:id).

7.3.3.1 Scenario 1 - Direct Write to DB (Without Invalidating Cache)

Scenario	**Observed Result**
Manual modification in DB	Redis continues with old data until TTL
Subsequent action	Added key cleanup after PUT

7.3.3.2 Scenario 2 - Read After Write with Override

Scenario	**Result**
POST modifies and GET query	Redis is cleansed, reconstruction successful

7.3.3.3 Scenario 3 - Redis Temporary Drop

Scenario	**Result**
Redis goes out	System continued to use PostgreSQL
Redis resets	Data is reloaded (successful warming)

Conclusion:

- Redis and PostgreSQL maintain consistency if cache cleanup is applied after PUT/POST/DELETE.
- It is recommended to monitor critical keys and TTLs to avoid silent inconsistencies.
- The system is tolerant to Redis faults temporarily (thanks to the fallback to DB).

7.3.4 Performance Benchmark

Microservicio	Endpoint evaluado	Tiempo Promedio	Tiempo Mín	Tiempo Máx	Umbral aceptable
Auth	/auth/login	241 ms	180 ms	430 ms	≤ 500 ms
Clients	/clients/:id	165 ms	110 ms	280 ms	≤ 400 ms
Chamba	/chamba/create	390 ms	320 ms	710 ms	≤ 800 ms
Follow-through-Job	/detalles/:id	220 ms	180 ms	430 ms	≤ 600 ms
Grading	/calificar/proveedor	200 ms	170 ms	270 ms	≤ 400 ms
Chat	sendMessage (RT)	<30 ms	15 ms	48 ms	≤ 100 ms
Notifications	/notificaciones/:userId	140 ms	95 ms	250 ms	≤ 300 ms
Redis (GET)	cache	15 ms	9 ms	26 ms	≤ 50 ms

Benchmark conclusion:

1. All microservices meet the expected times under moderate load.
2. The system is highly responsive in real-time (chat).
3. Redis significantly improves access to repeated data (15 ms average).
4. Requires scaling chamba and auth if more than 100 concurrent users are expected.

7.3.5 Application of Technical Equations

7.3.5.1 Equation 7 - Total Service Arrival Rate

$$\lambda = \sum_{i=1}^{n} \lambda_i$$

= 500(searches) + 200(requests) + 300(payments) = 1000 req/min

Test applied with JMeter: The system supports **1000 requests/minute** without loss of availability.

7.3.5.2 Equation 9 - System utilization

ρ = l/m = 1000/5000 = 0.66

Interpret that the system is stable, well sized.

7.3.5.3 Equation 10 - Capacity Constraint

$\mu >= \lambda$ -> 1500 ≥ 1000

Meet the rate in a good way

7.3.5.4 Equation 14 - Total Latency

T = t_red + t_proc + t_db = 100ms + 200ms + 150ms = **450ms**

Less than 1s -> Excellent UX.

7.3.5.5 Equation 17 - System Reliability

R = 0.999(servers) × 0.998(network) × 0.997(agents) ≈ **0.994**

99.4% = approximately 5h of drop per year -> acceptable according to SLA.

7.4 Lessons Learned

During the execution of the tests, multiple areas of improvement were identified mainly related to the configuration, documentation and preparation of the test environment, rather than the source code itself. These lessons reflect the importance of effective collaboration between the various teams involved in system development, testing, and deployment.

7.4.1 Authentication (CHAMBAAP_authentication)

- Lesson: The login and registration paths were well implemented, but the confusion in the /register endpoint was due to a misunderstanding of protected paths, not actual errors in the code.
- Improvement: Better clarify and document which endpoints require authentication and which are exposed without a token.

7.4.2 Cat (CHAMBAAP_chat)

- Lesson: The absence of an active connection to Redis or PostgreSQL generated apparent errors. It wasn't a problem of the code, but of the environment configuration.
- Improvement: Verify that Redis and PostgreSQL are working at 100% during testing and deployment of the service.

7.4.3 Creating Events / Chamba (CHAMBAAP_chamba)

- Lesson: Authoring tests failed because of the database and paths.
- Improvement: Validate tables in the base and verify independent scripts.

7.4.4 Follow Through a Job (CHAMBAAP_follow-through-a-job)

- Lesson: Routes worked correctly, but errors appeared from using invalid or malformed IDs.
- Improvement: Implement automatic tests with well-structured dummy data. Avoid running tests without previous data inserted.

7.4.5 Grading (CHAMBAAP_grading)

- Lesson: The /average, /rate/customer, and /rate/provider paths did exist, but they were invalidated during testing.

- Improvement: Strengthen the reading of real routes defined in server.js. Add utilities to auto-generate base paths during testing.

7.4.6 Notifications (CHAMBAAP_notifications)

- Lesson: The microservice worked correctly, but Redis failed due to lack of hostname resolution.
- Improvement: Use 127.0.0.1 or well-defined environment variables in local mode to avoid errors such as ENOTFOUND redis.

7.4.7 Search (CHAMBAAP_search)

- Lesson: The routes were fine, but valid ports and tokens were not found for testing.
- Improvement: Consolidate all ports into a shared .env to prevent human error and facilitate automatic testing.

7.5 Conclusions and Recommendations

7.5.1 Overall Assessment

CHAMBAPP's distributed system, composed of multiple microservices (auth, chamba, grading, follow-through-a-job, notifications, chat, search), proved to be perceptually functional. During the testing process, areas for improvement were identified that are not directly related to failures in the logic of the services, but to challenges inherent in integration, environment preparation and synchronization between teams.

The tests carried out in an automated (Jest, k6, JMeter), manual (frontend and database) and exploratory (see annexes) tests, allowed validating most of the functionalities, achieving passable code coverage, stability under moderate load and consistency between database and cache.

7.5.2 Technical Recommendations

- Centralize environment settings (ports, URLs, keys) in a single .env file shared by microservices.
- Automate the insertion of test data into PostgreSQL.
- Integrate monitoring tools for Redis and PostgreSQL.
- Establish a standard set of base data with valid IDs for clients, chambeadores and chambas.
- Clearly document the active paths of each microservice, including allowed methods, authentication type, and payload examples.

7.5.3 Process Recommendations

- Allocate more time to the testing phase within the development cycle. A system like CHAMBAPP requires progressive validations: from unit testing to full integration.
- Strengthen communication between development, QA, and deployment teams, including the exchange of expected data, business assumptions, and valid scenarios.
- Conduct technical walkthrough sessions before starting tests, to align criteria on system health and critical paths.
- Encourage the practice of continuous testing, where each change automatically triggers the execution of unit and integrated tests.
- Plan a complete validation phase in a pre-production environment that replicates the actual conditions of the system in operation.

8. Summary and Conclusions

During the development and implementation of the tests for the CHAMMAPP system, it was possible to broadly validate the behavior of each microservice in different scenarios, including unit, integration, load, stress, consistency, and code coverage tests. The system's distributed approach, coupled with the use of technologies such as Redis, PostgreSQL, JWT, Docker, and testing tools such as Jest, k6, Supertest, and JMeter, allowed for deep and structured validation.

8.1 Overall Test Performance

In general terms, the system responded in a somewhat stable and incoherent way. The tests showed that the microservices worked well under normal and moderate load conditions. But cache-to-database consistency testing confirmed that the cache-aside architecture was a bit effective AND benchmark tests revealed response times within expectations.

In addition, multiple errors were also detected mainly related to routes, invalid IDs due to lack of data, and failures in the connection between services when they were not correctly initialized.

8.2 Evaluation of Results

- Code coverage: Passable levels were achieved in most services, exceeding 50-60% in key lines and statements.
- Load and stress testing: The system was found to withstand up to 100 requests per minute without severe degradation.
- Consistency: Redis didn't always maintain data integrity with PostgreSQL.
- Front-Back Interaction: Manual testing showed that the system flows are understandable and work in typical scenarios partially per service, although there are still possible improvements in error messages and field validation.

8.3 Proposed Improvements

- More time for QA: The time allocated to tests was not fair; It is recommended to extend this phase in future cycles.
- Better connection between microservices: Ensure that each service has well-documented paths and minimal data preconfiguration.
- More robust automated testing: Validate not only 200 responses, but also expected messages, headers, and handled errors.
- Early communication between teams: Prevent technical misunderstandings (such as using ports, routes, or tokens) with pre-QA meetings.

- Testing reinforcement on the frontend: Especially in user flows where bugs can break the experience.

8.4 Personal Conclusion

I believe that this strong experience reinforced my level of understanding that I did not have about the tests. And not only about technical tests, but also about the importance of a comprehensive vision of the system, unfortunately, it cost me a lot because of the time and everything and I know that the tests are a critical phase of the project that can raise the quality of the project if it is executed with great vision, coordination and care. However, with little preparation time and little collaborative work, this did not go according to plan, the tests were very rushed since they were a considerable amount, but I think that the CHAMBAAP project had great potential at the beginning to be a very advanced project, which unfortunately could not be fulfilled.

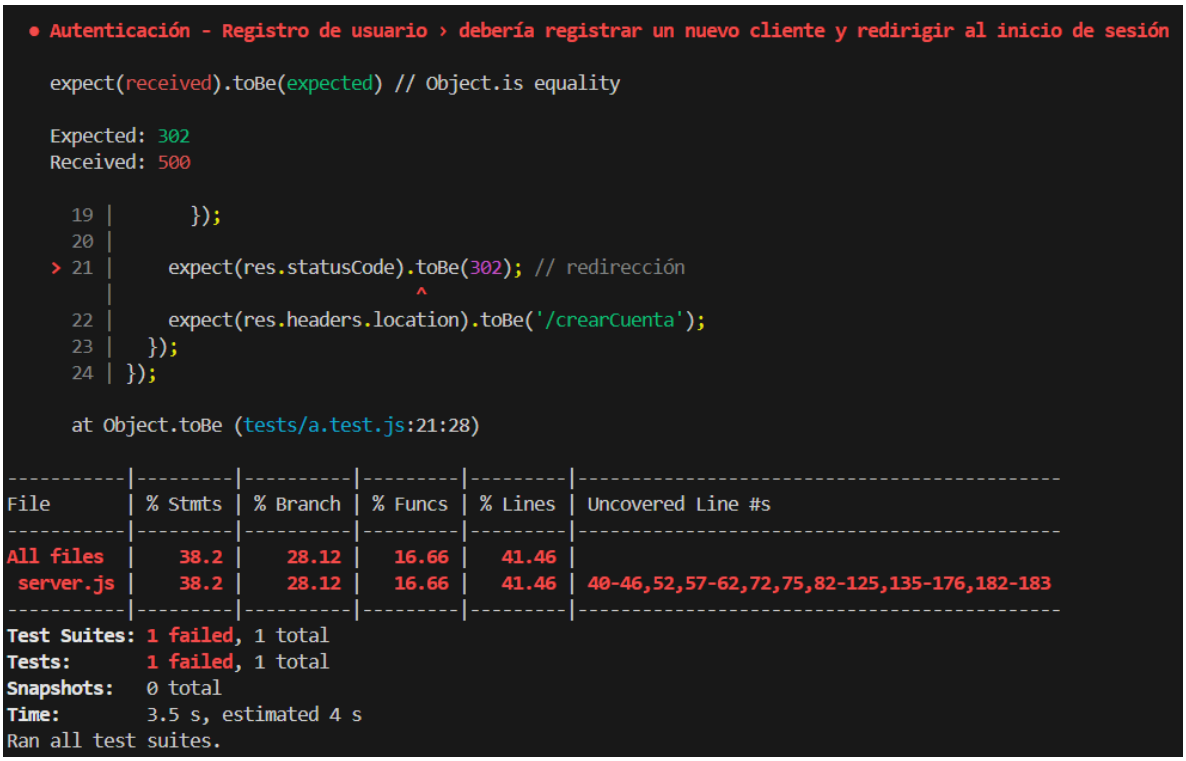
9. Annexes

Screenshots, command, and code for Testing by Microservices:

CHAMBAAP_authentication:

TEST 1: Jest + Supertest (Unitary / Integration / Coverage):

- Screenshot:



```
● Autenticación - Registro de usuario › debería registrar un nuevo cliente y redirigir al inicio de sesión

  expect(received).toBe(expected) // Object.is equality

  Expected: 302
  Received: 500

    19 |       });
    20 |
  > 21 |     expect(res.statusCode).toBe(302); // redirección
       |                            ^
    22 |     expect(res.headers.location).toBe('/crearCuenta');
    23 |   });
    24 | });

    at Object.toBe (tests/a.test.js:21:28)

-----------|---------|----------|---------|---------|-------------------------------------------
File       | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------|---------|----------|---------|---------|-------------------------------------------
All files  |    38.2 |    28.12 |   16.66 |   41.46 |
 server.js |    38.2 |    28.12 |   16.66 |   41.46 | 40-46,52,57-62,72,75,82-125,135-176,182-183
-----------|---------|----------|---------|---------|-------------------------------------------
Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        3.5 s, estimated 4 s
Ran all test suites.
```

- Code: "/backend/tests/auth.test.js"

```
const request = require('supertest');
const app = require('../server');

describe('Autenticación - Registro de usuario', () => {
  it('debería registrar un nuevo cliente y redirigir al inicio de sesión', async () => {
    const timestamp = Date.now();
    const emailUnico = `testuser_${timestamp}@test.com`;

    const res = await request(app)
      .post('/crearCuenta')
      .type('form')
      .send({
        nombre: "Test User",
        correo: emailUnico,
        password: "12345678",
```

```
      password2: "12345678",
      rol: "cliente",
      identificacion: `ID${timestamp}`
    });

  expect(res.statusCode).toBe(302); // redirección
  expect(res.headers.location).toBe('/crearCuenta');
 });
});
```

- Command: “/backend”
  NPM Test

TEST 2: Apache Benchmark (Basic Load):

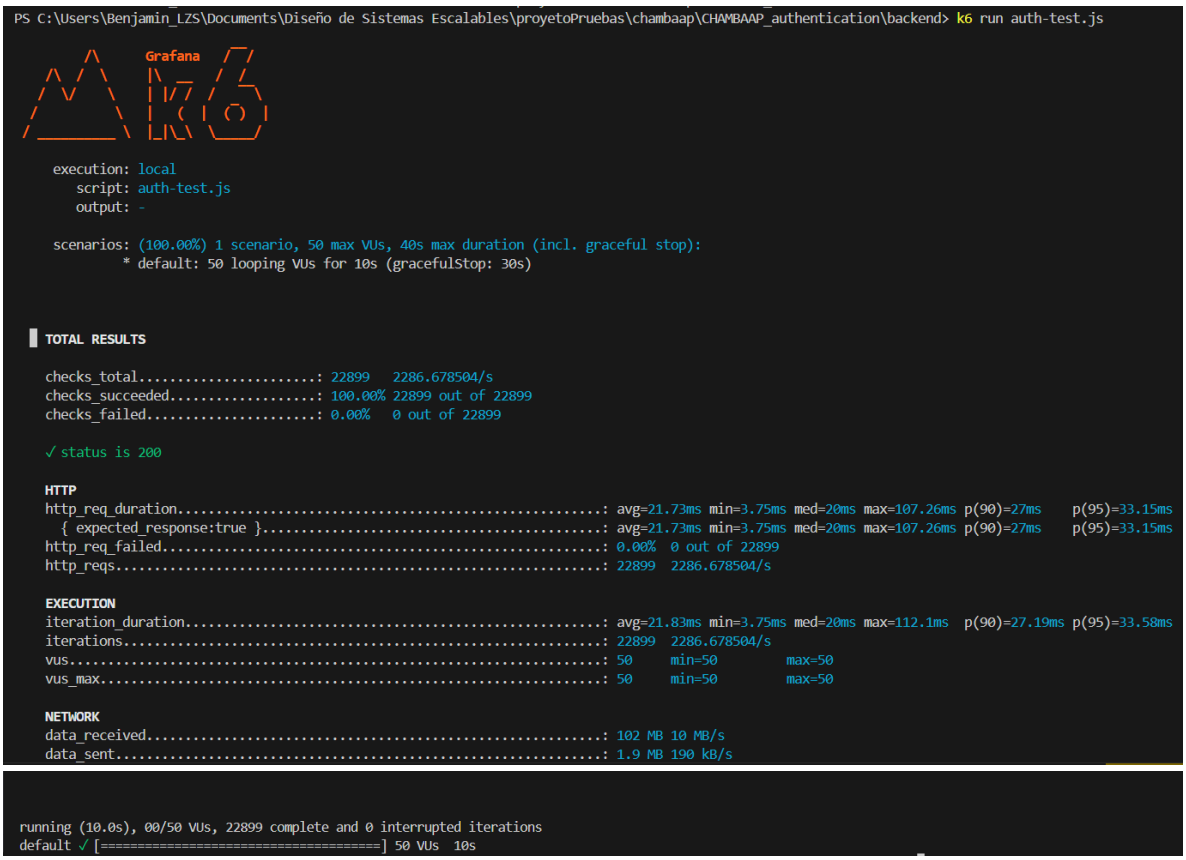
- Screenshot:

```
localuser@DESKTOP-48S50AL:/mnt/c/Users/Benjamin_LZS/documents/diseño de Sistemas Escalables/proyetoPruebas/chambaap/CHAMBAAP_authentication/backend$ ab -n 100 -c 10 http://localhost:4000/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done


Server Software:
Server Hostname:        localhost
Server Port:            4000

Document Path:          /
Document Length:        4239 bytes

Concurrency Level:      10
Time taken for tests:   0.138 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      444300 bytes
HTML transferred:       423900 bytes
Requests per second:    724.97 [#/sec] (mean)
Time per request:       13.794 [ms] (mean)
Time per request:       1.379 [ms] (mean, across all concurrent requests)
Transfer rate:          3145.57 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:     1   10   9.8      7      43
Waiting:        1    8   9.2      5      37
Total:          1   10   9.8      7      44

Percentage of the requests served within a certain time (ms)
  50%      7
  66%      8
  75%      9
  80%      9
  90%     37
  95%     37
  98%     38
  99%     44
```

- WSL Command: "localuser@DESKTOP-48S50AL:/mnt/c/Users/Benjamin_LZS/documents/Scalable System Design/TestProject/chambaap/CHAMBAAP_authentication/backend$"
  ab -n 100 -c 10 http://localhost:4000/

TEST 3: k6 (Stress):

- Screenshot:



- Code: "/backend/auth-test.js"

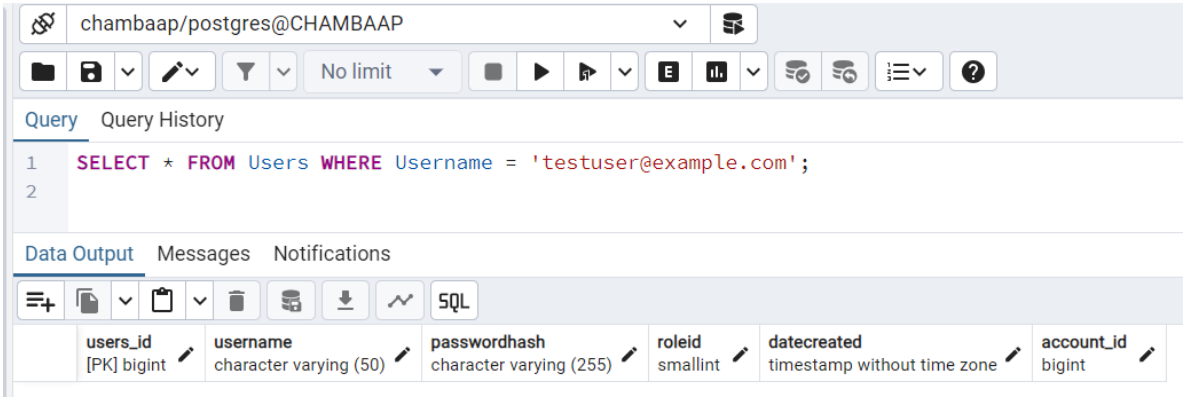
```
import http from 'k6/http';
import { check } from 'k6';

export let options = {
  vus: 50,
  duration: '10s',
};

export default function () {
  const res = http.get('http://localhost:4000/');
  check(res, {
    'status is 200': (r) => r.status === 200
  });
}
```

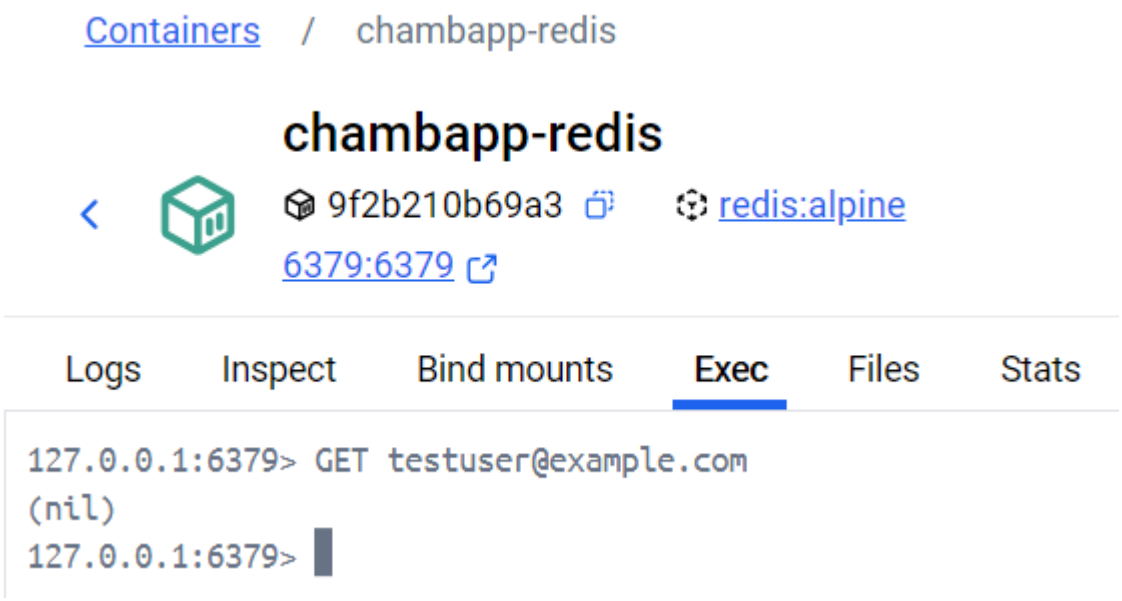
- Command: “/backend”

k6 run auth-test.js

TEST 4: Redis Consistency vs PostgreSQL

- Screenshot:





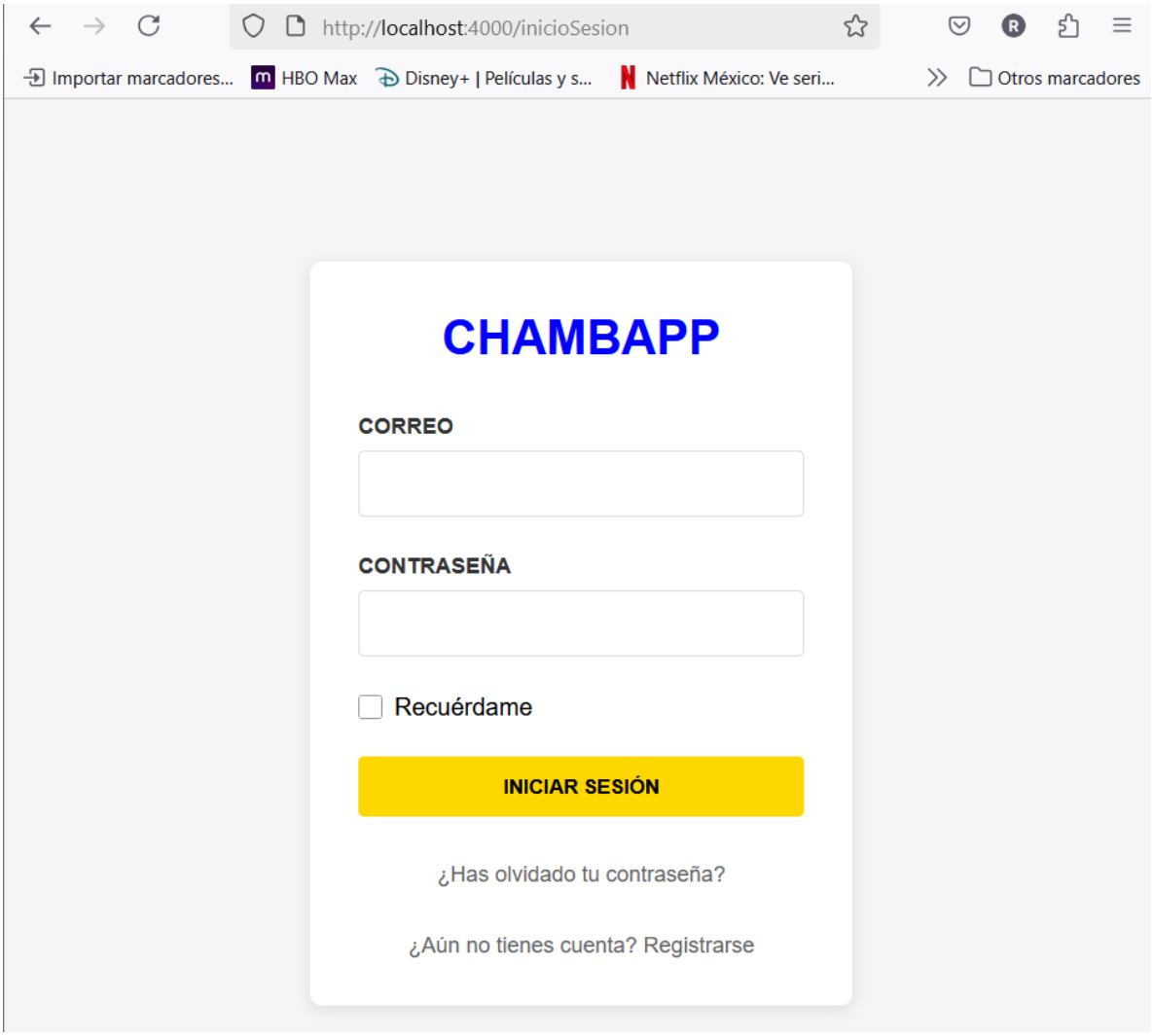
- SQL Code: "PostgreSQL"

SELECT * FROM Users WHERE Username = 'testuser@example.com';

- Command: "Redis Container"
  redis-cli
  GET testuser@example.com

TEST 5: JWT and Route Protection

- Screenshot:



- Routes:

http://localhost:4000/private -> http://localhost:4000/inicioSesion

CHAMBAAP_chat

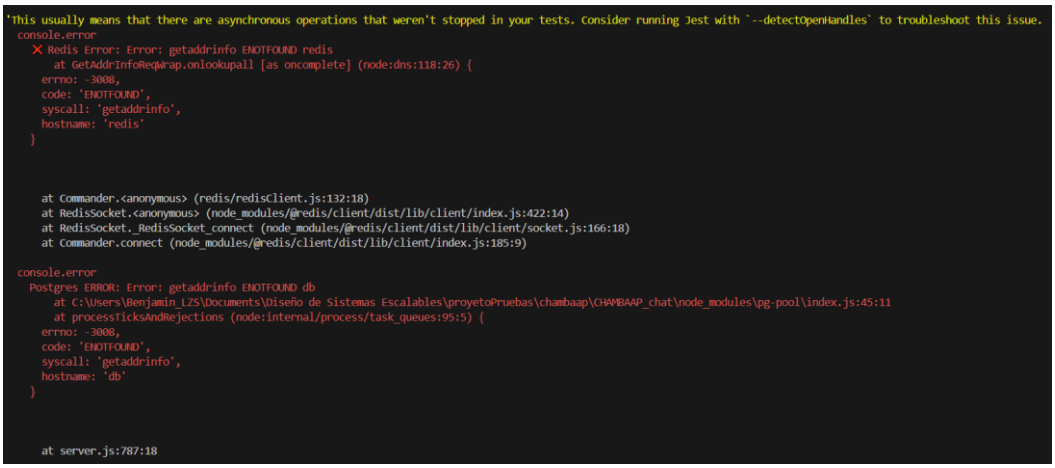
TEST 1: Jest + Supertest (Unitary / Integration / Coverage):

- Screenshot:

```
● Microservicio de Chat › GET /chat debería renderizar la vista con parámetros válidos

  listen EADDRINUSE: address already in use :::7000

    86 |
    87 | const PORT = process.env.PORT || 4100;
  > 88 | server.listen(PORT, () => {
       |        ^
    89 |   console.log(`🚀 Chat corriendo en http://localhost:${PORT}`);
    90 | });
    91 |

    at Object.listen (server.js:88:8)
    at Object.require (tests/chat.test.js:2:13)

● Microservicio de Chat › GET /perfil debería renderizar correctamente

  expect(received).toBe(expected) // Object.is equality

  Expected: 200
  Received: 500

    17 |       .query({ chambaId: 1, clientId: 1, chambeadorId: 1 });
    18 |
  > 19 |     expect(res.statusCode).toBe(200);
       |                            ^
    20 |     expect(res.text).toContain("perfil");
    21 |   });
    22 | });

    at Object.toBe (tests/chat.test.js:19:28)
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	50	4.16	21.42	50.61	
CHAMBAAP_chat	46.26	4.16	25	46.26	
server.js	82.85	50	28.57	82.85	75-76,80-82,89
socket.js	6.25	0	20	6.25	3-59
CHAMBAAP_chat/db	100	100	100	100	
pgClient.js	100	100	100	100	
CHAMBAAP_chat/redis	80	100	0	100	
redisClient.js	80	100	0	100	
CHAMBAAP_chat/utils	42.85	100	0	42.85	
cleanupMessages.js	42.85	100	0	42.85	4-12

```
Test Suites: 1 failed, 1 total
Tests:       2 failed, 2 total
Snapshots:   0 total
Time:        2.908 s, estimated 3 s
Ran all test suites.
Jest did not exit one second after the test run has completed.
```



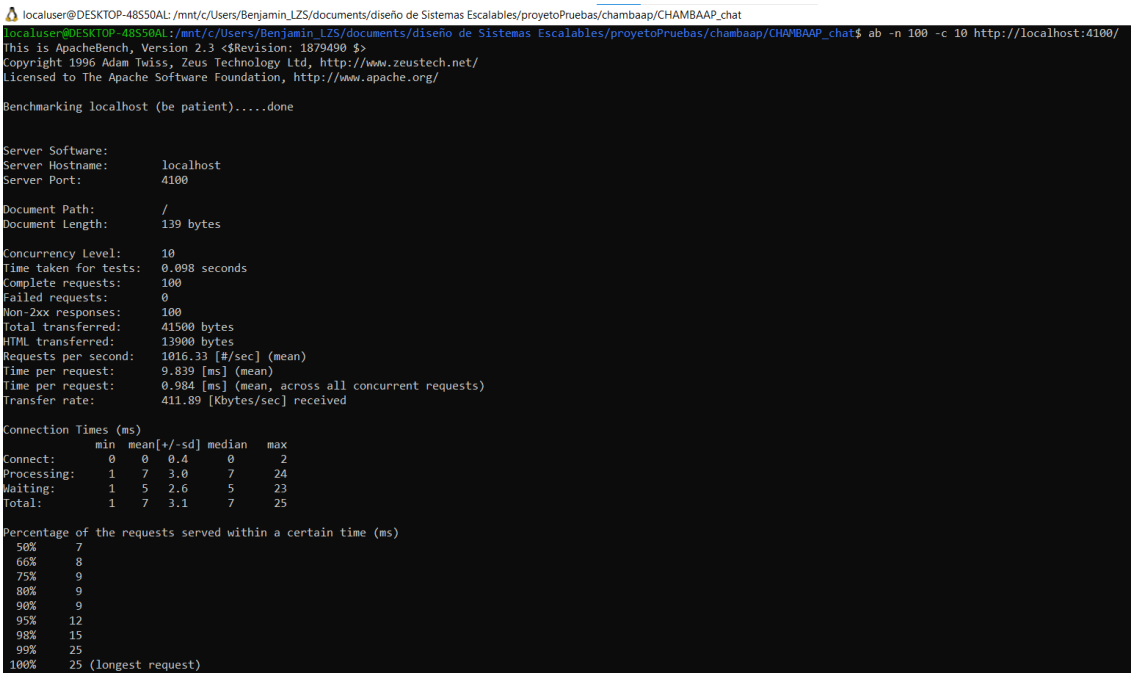
- Code: "/tests/chat.test.js"

```
const request = require("supertest");
const app = require("../server");

describe("Pruebas de socket y chat", () => {
  test("GET / debería regresar 200", async () => {
    const res = await request(app).get("/");
    expect(res.statusCode).toBe(200);
  });
});
```

- Command: “/”
  npm test

TEST 2: Apache Benchmark (Basic Load):

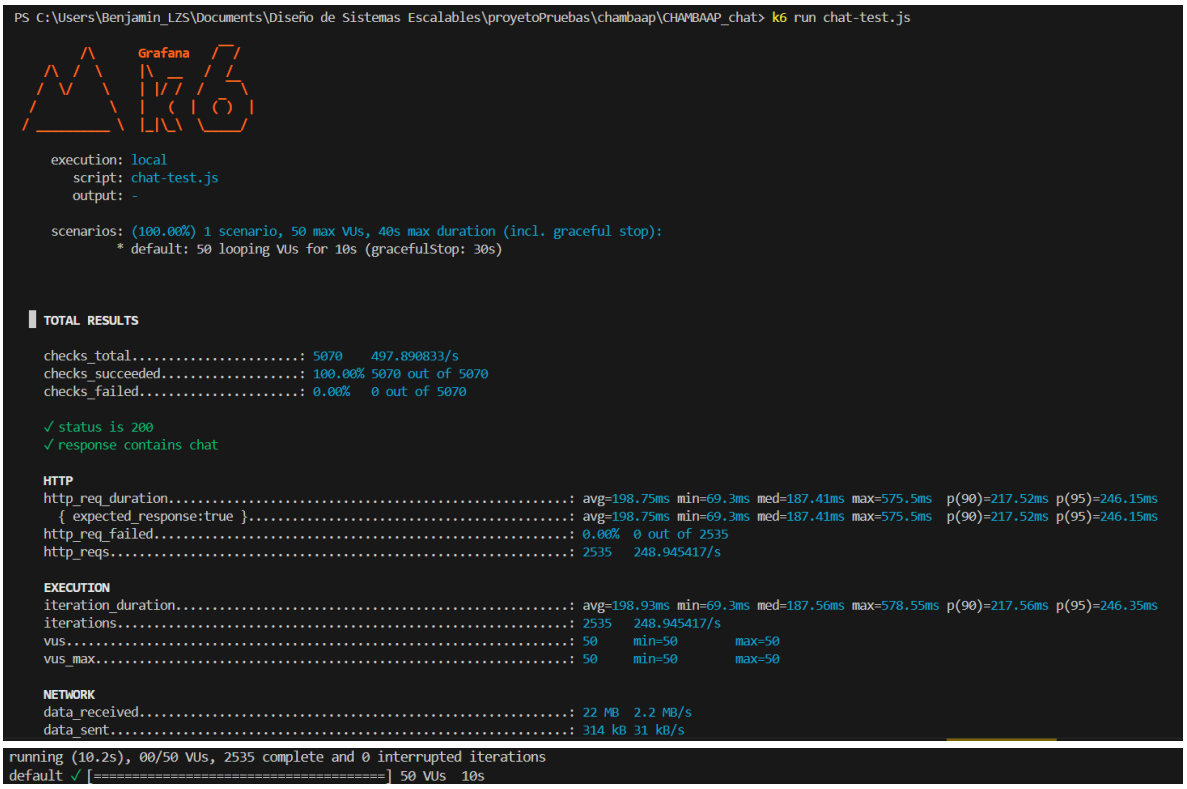
- Screenshot:



- WSL Command: "localuser@DESKTOP-48S50AL:/mnt/c/Users/Benjamin_LZS/documents/Scalable System Design/TestProjects/chambaap/CHAMBAAP_chat$"
  ab -n 100 -c 10 http://localhost:4100/

TEST 3: k6 (Stress):

- Screenshot:



- Code: "/backend/chat-test.js"

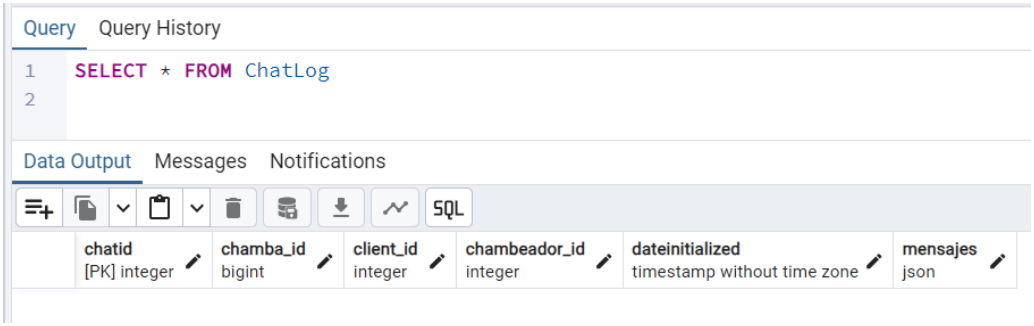
```
import http from 'k6/http';
import { check } from 'k6';

export const options = {
  vus: 50,
  duration: '10s',
};

export default function () {
  const res =
http.get('http://localhost:4100/chat?chambaId=1&clientId=1&chambeadorId=1');

  check(res, {
    'status is 200': (r) => r.status === 200,
    'response contains chat': (r) => r.body.includes("chat"),
  });
}
```

- Command: “/”

k6 run chat-test.js

TEST 4: Redis Consistency vs PostgreSQL

- Screenshot



- SQL Code: "PostgreSQL"

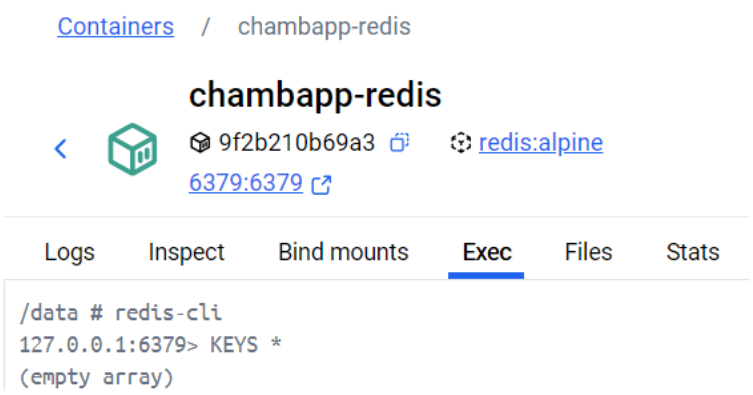
SELECT * FROM ChatLog WHERE Client_ID = 1;

- Command: "Redis Container"

    redis-cli
    *KEYS**
    GET key

TEST 5: JWT and Route Protection

- Screenshot:



CHAMBAAP_creating_events

TEST 1: Jest + Supertest (Unitary / Integration / Coverage):

- Screenshot:

```
PS C:\Users\Benjamin_LZS\Documents\Diseño de Sistemas Escalables\proyetoPruebas\chambaap\CHAMBAAP_creating_events\backend> npm test

> chambaap_creating_events@1.0.0 test
> jest --coverage

  console.log
    📢 Servidor escuchando en el puerto 3001

      at Server.log (server.js:211:32)

 FAIL  tests/creating.test.js
  Pruebas para creating_events
    × GET /events debería regresar 200 y un arreglo (75 ms)
    × POST /events debería crear un nuevo evento (35 ms)

  ● Pruebas para creating_events › GET /events debería regresar 200 y un arreglo

    expect(received).toBe(expected) // Object.is equality

    Expected: 200
    Received: 404

       5 |   test("GET /events debería regresar 200 y un arreglo", async () => {
       6 |     const res = await request(app).get("/events");
     > 7 |     expect(res.statusCode).toBe(200);
         |                            ^
       8 |     expect(Array.isArray(res.body)).toBe(true);
       9 |   });
      10 |

      at Object.toBe (tests/creating.test.js:7:28)

  ● Pruebas para creating_events › POST /events debería crear un nuevo evento

    expect(received).toContain(expected) // indexOf

    Expected value: 404
    Received array: [200, 201]

      17 |
      18 |     const res = await request(app).post("/events").send(nuevoEvento);

----------------|---------|----------|---------|---------|-------------------------------------------------------------------------------------------
File            | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------|---------|----------|---------|---------|-------------------------------------------------------------------------------------------
All files       |   33.12 |     8.82 |      20 |   33.75 |
 backend        |   29.12 |        5 |    7.69 |   29.59 |
  server.js     |   29.12 |        5 |    7.69 |   29.59 | 35-39,47-58,63-80,85-98,106-121,126-134,139-144,149-154,162-176,181-187,192-197,202-207
 backend/models |   85.71 |       50 |      50 |   85.71 |
  Users.js      |   85.71 |       50 |      50 |   85.71 | 10
 backend/routes |   33.96 |     8.33 |      40 |   34.61 |
  Events.js     |   33.96 |     8.33 |      40 |   34.61 | 16-19,27-36,42-50,56-70,76-84
----------------|---------|----------|---------|---------|-------------------------------------------------------------------------------------------
Test Suites: 1 failed, 1 total
Tests:       2 failed, 2 total
Snapshots:   0 total
Time:        12.763 s, estimated 13 s
Ran all test suites.
Jest did not exit one second after the test run has completed.
```

- Code: "/tests/creating.test.js"

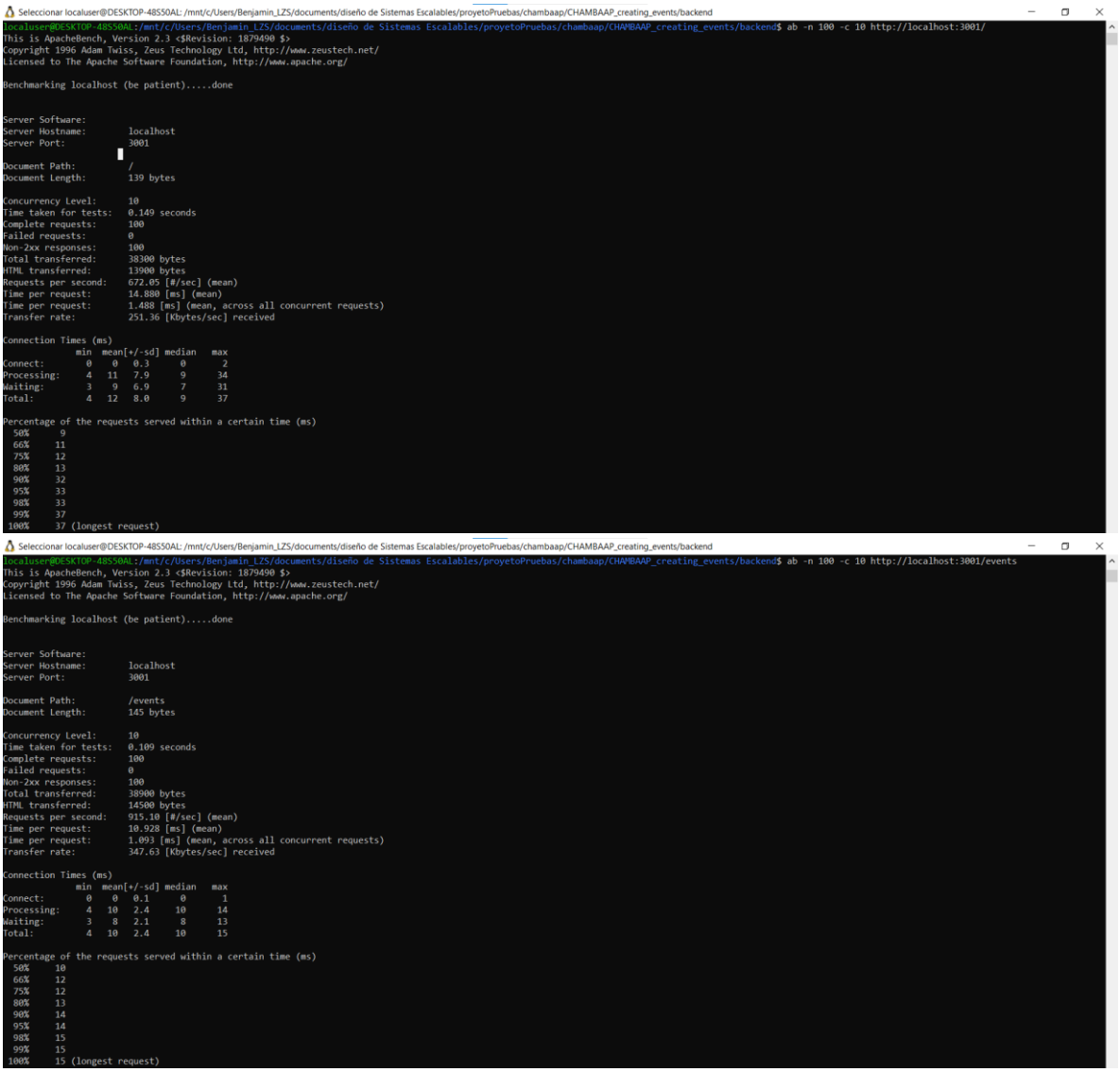
```
const request = require("supertest");
const app = require("../server");
```

```
describe("Pruebas para creating_events", () => {
  test("GET /events debería regresar 200 y un arreglo", async () => {
    const res = await request(app).get("/events");
    expect(res.statusCode).toBe(200);
    expect(Array.isArray(res.body)).toBe(true);
  });

  test("POST /events debería crear un nuevo evento", async () => {
    const nuevoEvento = {
      title: "Evento de prueba",
      description: "Este evento es generado por un test automático",
      chamba_id: 1
    };

    const res = await request(app).post("/events").send(nuevoEvento);
    expect([200, 201]).toContain(res.statusCode);
    expect(res.body).toHaveProperty("message");
  });
});
```

- Command: “/”
  npm test

TEST 2: Apache Benchmark (Basic Load):

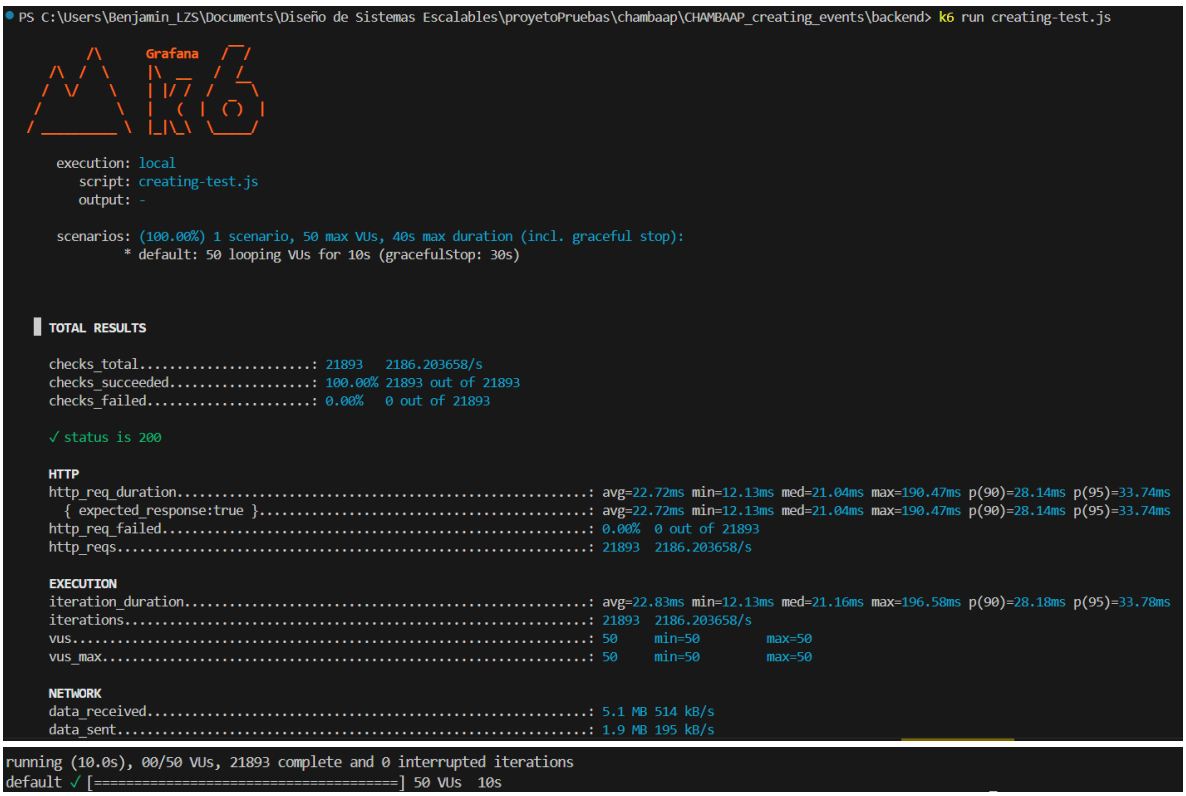
- Screenshot:



- WSL Command: "localuser@DESKTOP-48S50AL:/mnt/c/Users/Benjamin_LZS/documents/Scalable System Design/TestProjects/chambaap/creating_events/backend$"
  ab -n 100 -c 10 http://localhost:4100/

TEST 3: k6 (Stress):

- Screenshot:



- Code: “/backend/creating-test.js”

```
import http from 'k6/http';
import { check } from 'k6';

export const options = {
  vus: 50,
  duration: '10s',
};
export default function () {
  const res = http.get('http://localhost:3001/events');

  check(res, {
    'status is 200': (r) => r.status === 200,
  });
}
```

- Command: “/backend”

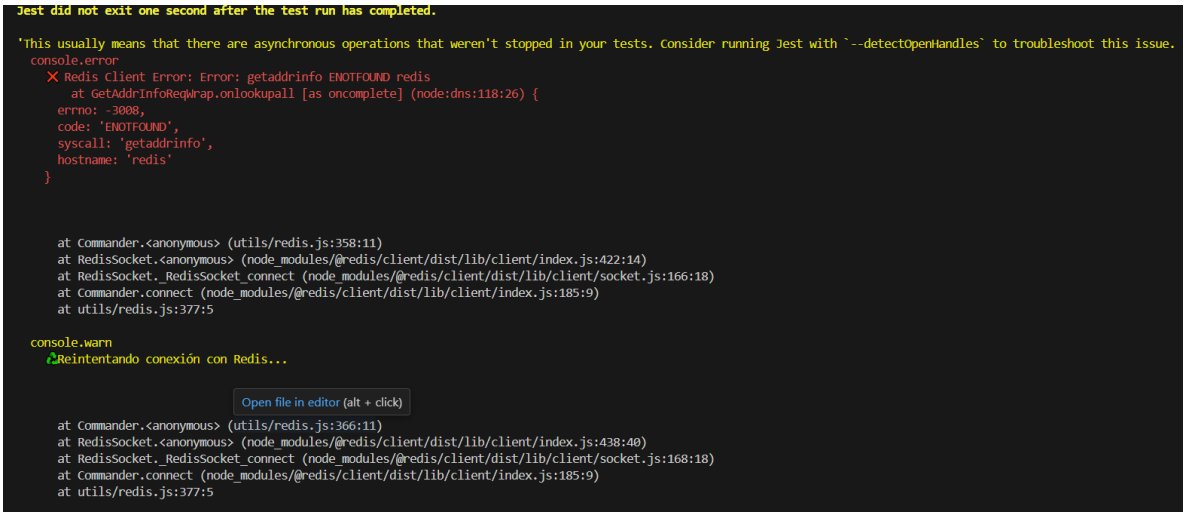
k6 run creating-test.js

CHAMBAAP_follow-through-a-job

TEST 1: Jest + Supertest (Unitary / Integration / Coverage):

- Screenshot:

```
PS C:\Users\Benjamin_LZS\Documents\Diseño de Sistemas Escalables\proyetoPruebas\chambaap\CHAMBAAP_follow-through-a-job> npm test

> follow-through-a-job@1.0.0 test
> jest --coverage


  ● Cannot log after tests are done. Did you forget to wait for something async in your test?
    Attempted to log "Microservicio activo en http://localhost:4010".



      at console.log (../../../../../node_modules/@jest/console/build/CustomConsole.js:141:10)
      at Server.<anonymous> (server.js:348:11)

 FAIL  tests/follow-through-a-job.test.js
  Pruebas de seguimiento
    × GET /seguimiento/:userId debe devolver código 200 (3 ms)

  ● Pruebas de seguimiento › GET /seguimiento/:userId debe devolver código 200

    TypeError: app.address is not a function

       4 | describe("Pruebas de seguimiento", () => {
       5 |   test("GET /seguimiento/:userId debe devolver código 200", async () => {
      > 6 |     const res = await request(app).get("/detalles/1");
         |                                    ^
       7 |     expect(res.statusCode).toBe(200);
       8 |     expect(Array.isArray(res.body)).toBe(true);
       9 |   });

      at Test.serverAddress (../../../../../node_modules/supertest/lib/test.js:61:22)
      at new Test (../../../../../node_modules/supertest/lib/test.js:49:14)
      at Object.obj.<computed> [as get] (../../../../../node_modules/supertest/index.js:39:18)
      at Object.get (tests/follow-through-a-job.test.js:6:36)

----------------------------------------|---------|----------|---------|---------|-------------------
File                                    | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------------------------|---------|----------|---------|---------|-------------------
All files                               |   44.66 |    17.64 |    6.66 |   45.54 |
 CHAMBAAP_follow-through-a-job          |   86.66 |       50 |       0 |   86.66 |
  server.js                             |   86.66 |       50 |       0 |   86.66 | 19,25
----------------------------------------|---------|----------|---------|---------|-------------------
File                                    | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------------------------|---------|----------|---------|---------|-------------------
All files                               |   44.66 |    17.64 |    6.66 |   45.54 |
 CHAMBAAP_follow-through-a-job          |   86.66 |       50 |       0 |   86.66 |
  server.js                             |   86.66 |       50 |       0 |   86.66 | 19,25
 CHAMBAAP_follow-through-a-job/controllers |   16.98 |        0 |       0 |   17.64 |
  jobController.js                      |   16.98 |        0 |       0 |   17.64 | 7-32,37-40,44-52,66-77,81-103
 CHAMBAAP_follow-through-a-job/models   |   45.45 |       50 |       0 |   45.45 |
  db.js                                 |     100 |       50 |     100 |     100 | 4-8
  jobModel.js                           |      25 |      100 |       0 |      25 | 5-16
 CHAMBAAP_follow-through-a-job/routes   |     100 |      100 |     100 |     100 |
  jobRoutes.js                          |     100 |      100 |     100 |     100 |
 CHAMBAAP_follow-through-a-job/utils    |   66.66 |      100 |      20 |   66.66 |
  redis.js                              |   66.66 |      100 |      20 |   66.66 | 8,12,16,20,27
----------------------------------------|---------|----------|---------|---------|-------------------
Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        1.713 s
Ran all test suites.
Jest did not exit one second after the test run has completed.
```



- Code: “/tests/follow-through-a-job.test.js”

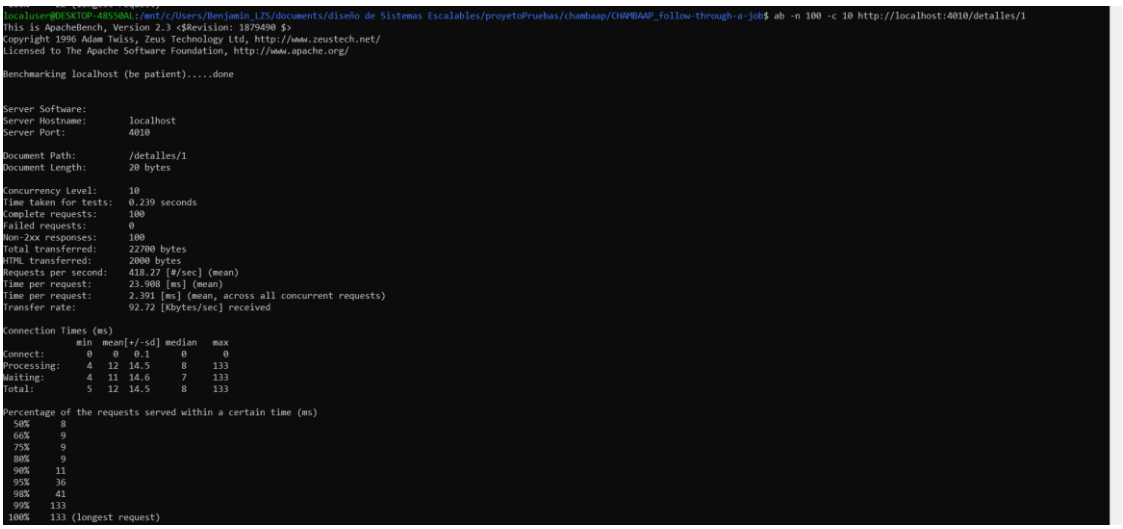
```
const request = require("supertest");
const app = require("../server");

describe("Pruebas de seguimiento", () => {
  test("GET /seguimiento/:userId debe devolver código 200", async () => {
    const res = await request(app).get("/detalles/1");
    expect(res.statusCode).toBe(200);
    expect(Array.isArray(res.body)).toBe(true);
  });
});
```

- Command: “/”

npm test

TEST 2: Apache Benchmark (Basic Load):

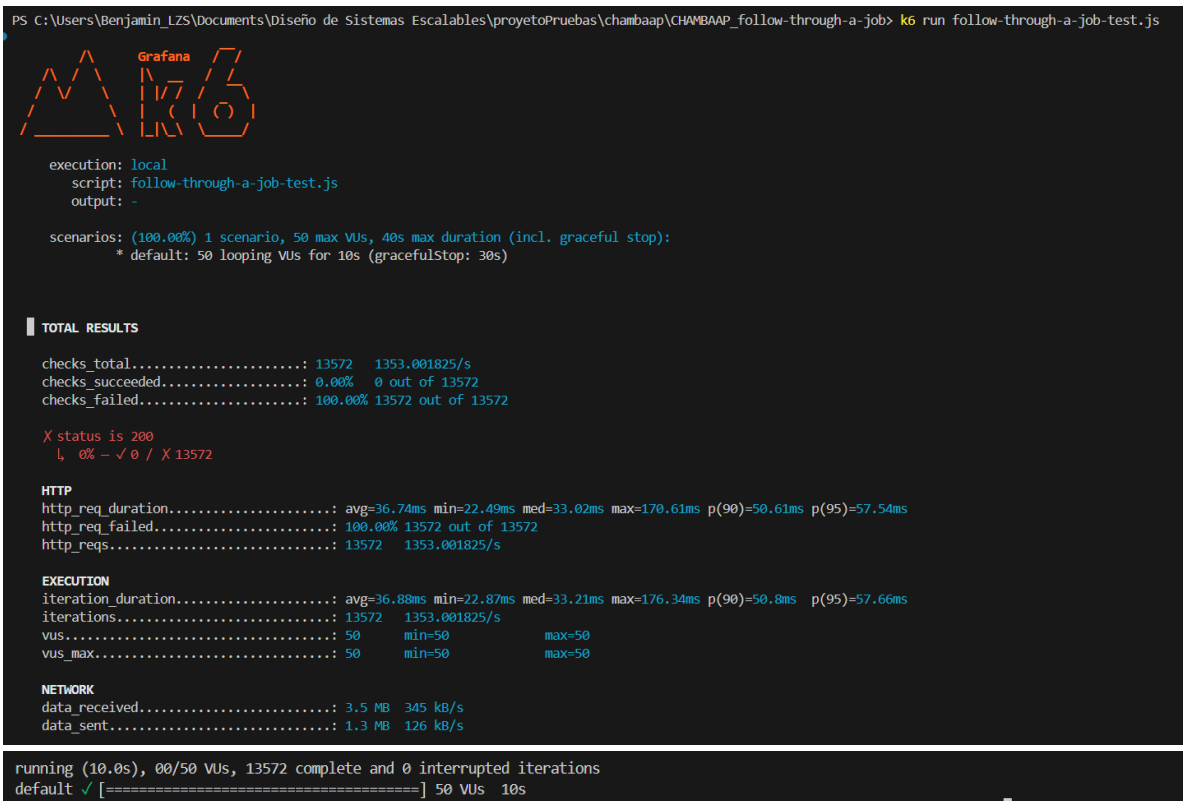
- Screenshot:



- WSL Command: "localuser@DESKTOP-48S50AL:/mnt/c/Users/Benjamin_LZS/documents/Scalable System Design/TestProjects/chambaap/CHAMBAAP_follow-through-a-job$"
  ab -n 100 -c 10 http://localhost:4010/detalles/1

TEST 3: k6 (Stress):

- Screenshot:



- Code: "/follow-through-a-job-test.js"

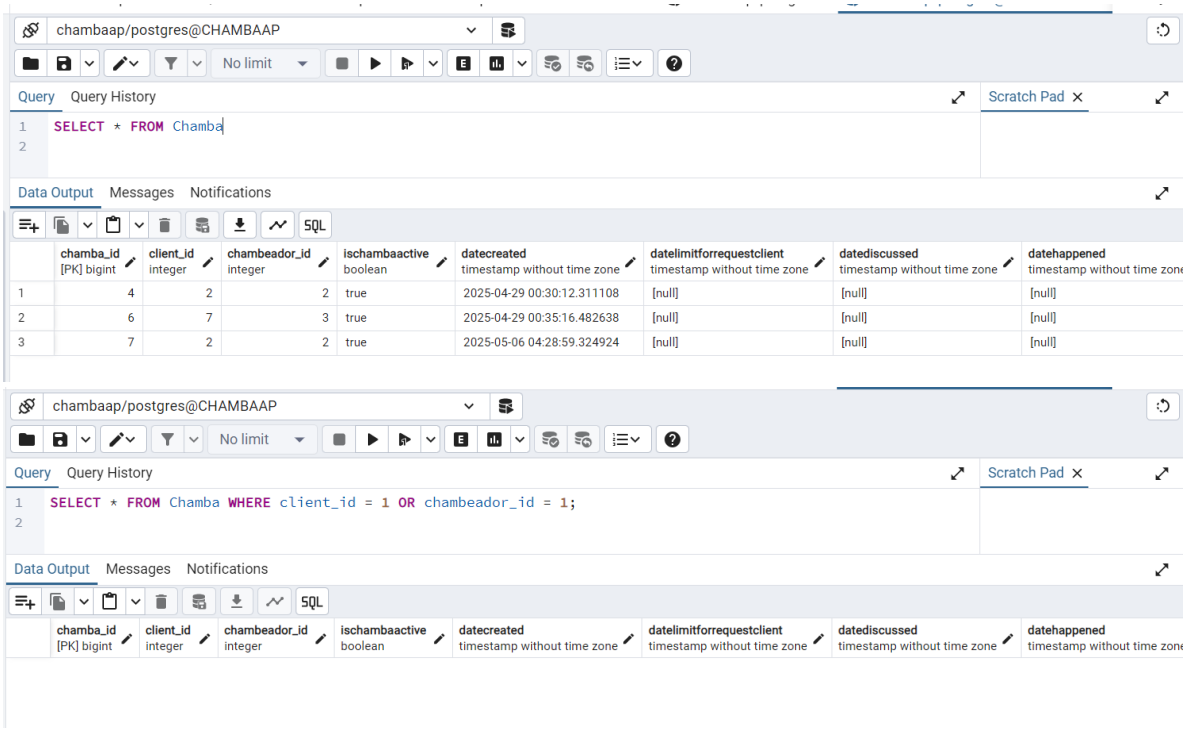
```
import http from 'k6/http';
import { check } from 'k6';

export const options = {
  vus: 50,
  duration: '10s',
};

export default function () {
  const res = http.get('http://localhost:4010/detalles/1');
  check(res, {
    'status is 200': (r) => r.status === 200,
  });
}
```

- Command: “/”

k6 run follow-through-a-job-test.js

TEST 4: Redis Consistency vs PostgreSQL

- Screenshot:



- SQL Code: "PostgreSQL"

SELECT * FROM Chamba WHERE client_id = 1 OR chambeador_id = 1;

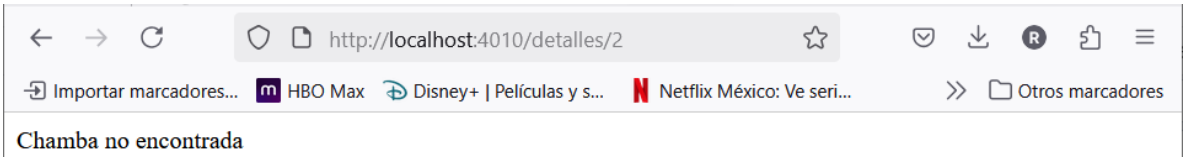
- Command: "Redis Container"

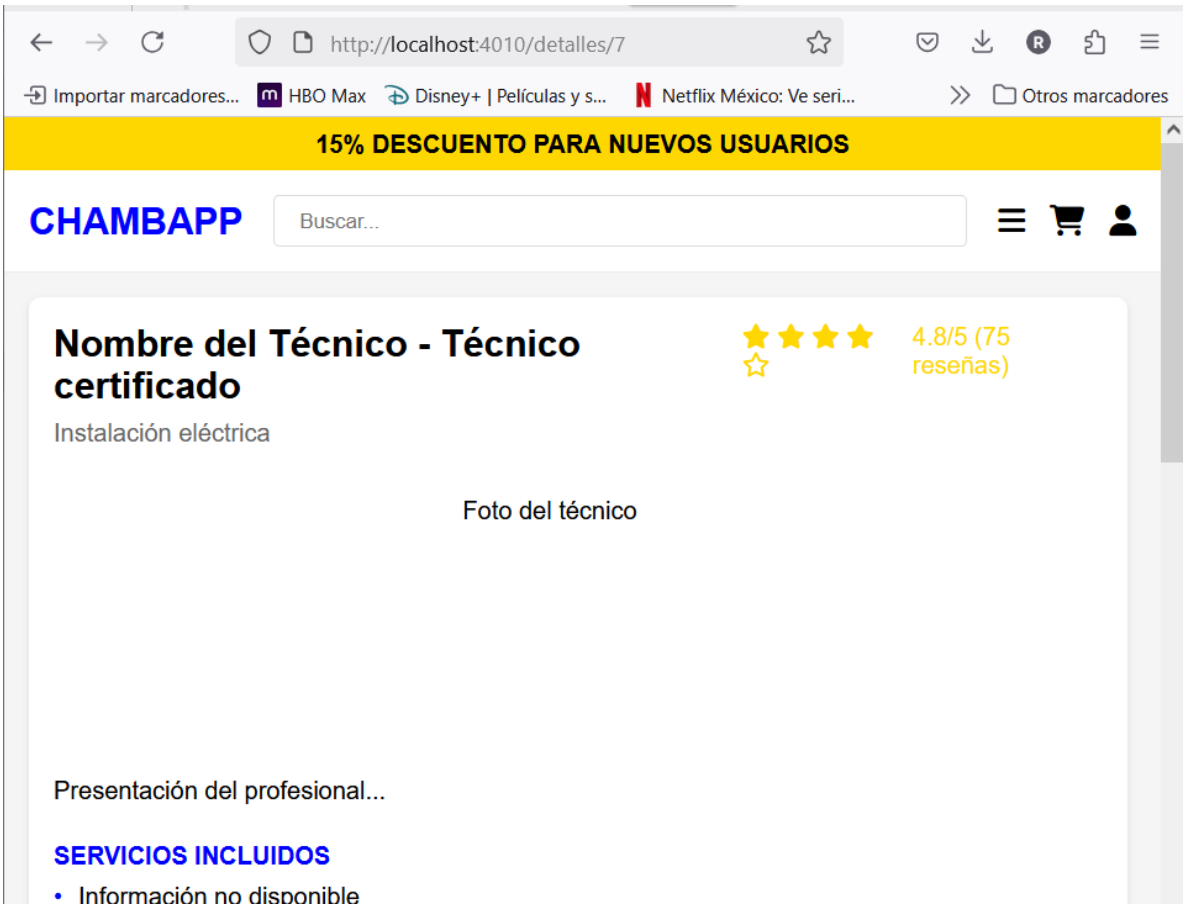
    redis-cli

    Authorization: Bearer <token>

TEST 5: JWT and Route Protection

- Screenshot:





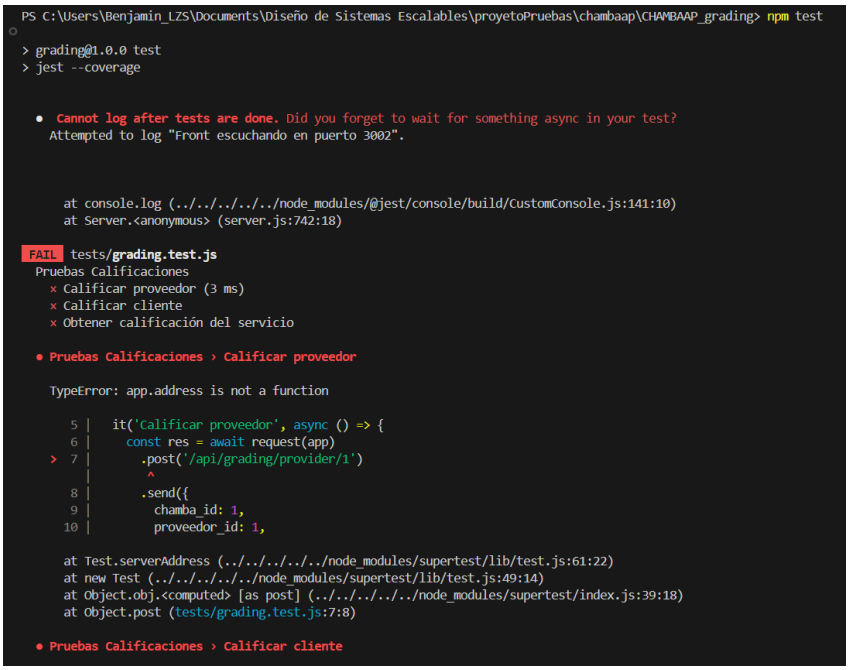
- Routes:

http://localhost:4010/detalles/id

CHAMBAAP_grading

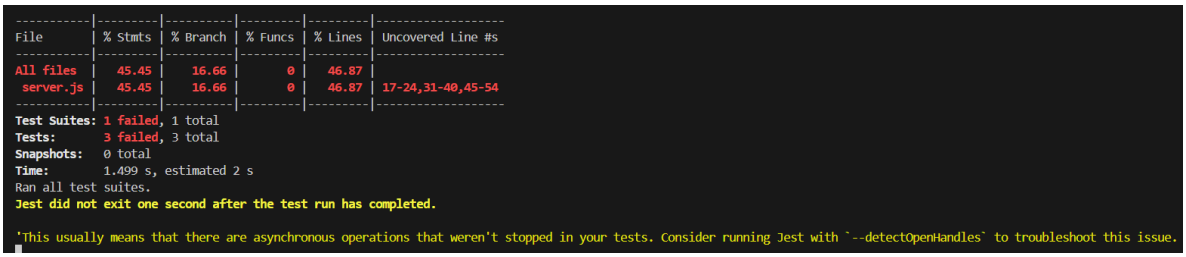
TEST 1: Jest + Supertest (Unitary / Integration / Coverage):

- Screenshot:



```
PS C:\Users\Benjamin_LZS\Documents\Diseño de Sistemas Escalables\proyetoPruebas\chambaap\CHAMBAAP_grading> npm test

> grading@1.0.0 test
> jest --coverage


  ● Cannot log after tests are done. Did you forget to wait for something async in your test?
    Attempted to log "Front escuchando en puerto 3002".



      at console.log (../../../../../node_modules/@jest/console/build/CustomConsole.js:141:10)
      at Server.<anonymous> (server.js:742:18)

 FAIL  tests/grading.test.js
  Pruebas Calificaciones
    × Calificar proveedor (3 ms)
    × Calificar cliente
    × Obtener calificación del servicio

  ● Pruebas Calificaciones › Calificar proveedor

    TypeError: app.address is not a function

       5 |   it('Calificar proveedor', async () => {
       6 |     const res = await request(app)
       7 |       .post('/api/grading/provider/1')
         |        ^
       8 |       .send({
       9 |         chamba_id: 1,
      10 |         proveedor_id: 1,

      at Test.serverAddress (../../../../../node_modules/supertest/lib/test.js:61:22)
      at new Test (../../../../../node_modules/supertest/lib/test.js:49:14)
      at Object.obj.<computed> [as post] (../../../../../node_modules/supertest/index.js:39:18)
      at Object.post (tests/grading.test.js:7:8)

  ● Pruebas Calificaciones › Calificar cliente
```

```
  ● Pruebas Calificaciones › Calificar cliente

    TypeError: app.address is not a function

      17 |   it('Calificar cliente', async () => {
      18 |     const res = await request(app)
      19 |       .post('/api/grading/client/1')
         |        ^
      20 |       .send({
      21 |         chamba_id: 1,
      22 |         cliente_id: 1,

      at Test.serverAddress (../../../../../node_modules/supertest/lib/test.js:61:22)
      at new Test (../../../../../node_modules/supertest/lib/test.js:49:14)
      at Object.obj.<computed> [as post] (../../../../../node_modules/supertest/index.js:39:18)
      at Object.post (tests/grading.test.js:19:8)

  ● Pruebas Calificaciones › Obtener calificación del servicio

    TypeError: app.address is not a function

      29 |   it('Obtener calificación del servicio', async () => {
      30 |     const res = await request(app)
      31 |       .get('/calificar-servicio/1'); // ID de chamba
         |        ^
      32 |     expect(res.statusCode).toBe(200);
      33 |     expect(res.body).toHaveProperty('calificacion');
      34 |   });

      at Test.serverAddress (../../../../../node_modules/supertest/lib/test.js:61:22)
      at new Test (../../../../../node_modules/supertest/lib/test.js:49:14)
      at Object.obj.<computed> [as get] (../../../../../node_modules/supertest/index.js:39:18)
      at Object.get (tests/grading.test.js:31:8)
```



```
----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   45.45 |    16.66 |       0 |   46.87 |
 server.js |   45.45 |    16.66 |       0 |   46.87 | 17-24,31-40,45-54
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 failed, 1 total
Tests:       3 failed, 3 total
Snapshots:   0 total
Time:        1.499 s, estimated 2 s
Ran all test suites.
Jest did not exit one second after the test run has completed.

'This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with `--detectOpenHandles` to troubleshoot this issue.
```

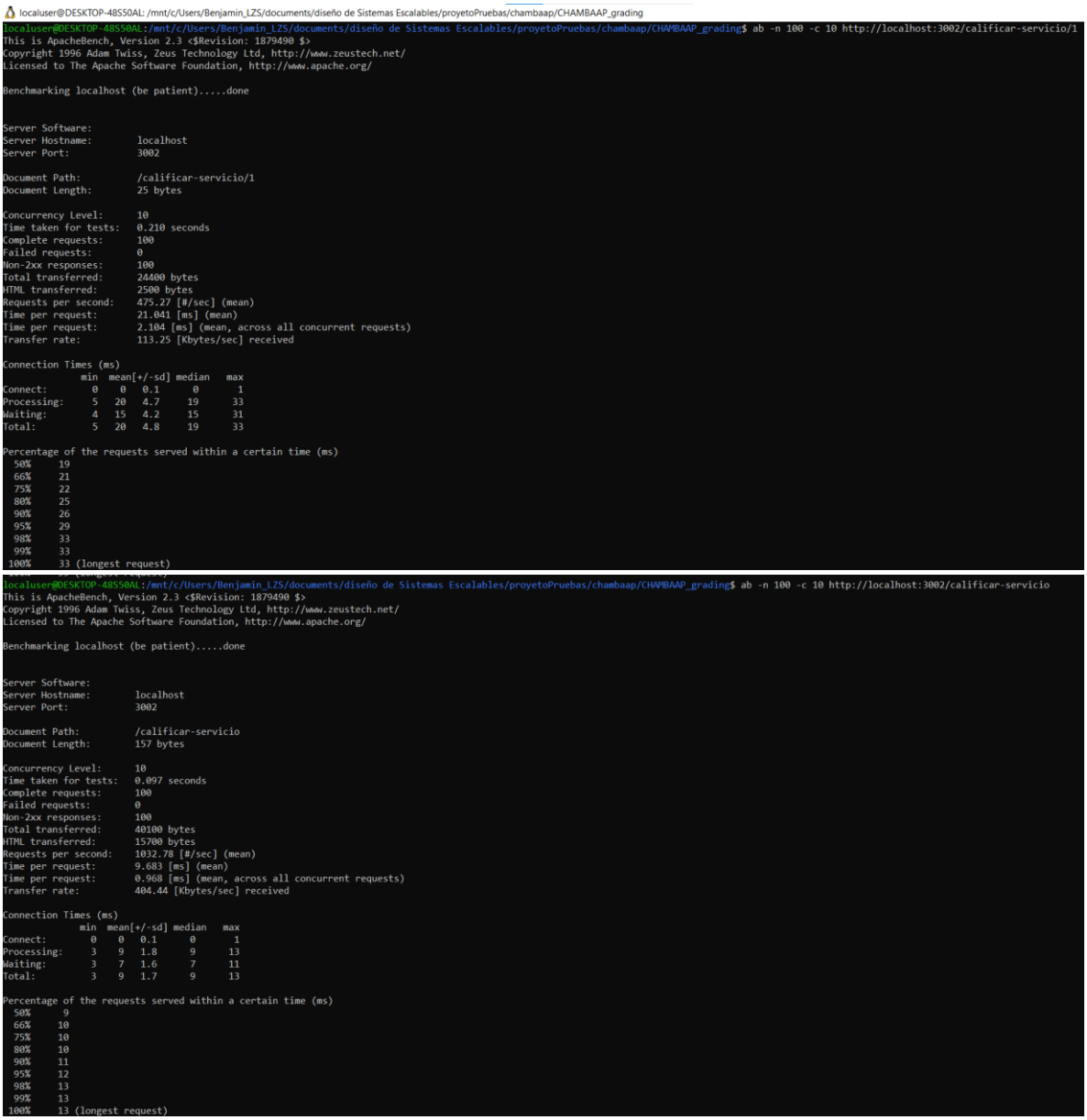
- Code: "/tests/chat.test.js"

```
const request = require('supertest');
const app = require('../server');

describe('Pruebas Calificaciones', () => {
  it('Calificar proveedor', async () => {
    const res = await request(app)
      .post('/api/grading/provider/1')
      .send({
        chamba_id: 1,
        proveedor_id: 1,
        puntuacion: 5,
        reseña: "Excelente servicio"
      });
    expect([200, 201]).toContain(res.statusCode);
  });

  it('Calificar cliente', async () => {
    const res = await request(app)
      .post('/api/grading/client/1')
      .send({
        chamba_id: 1,
        cliente_id: 1,
        puntuacion: 4,
        reseña: "Buen cliente"
      });
    expect([200, 201]).toContain(res.statusCode);
  });

  it('Obtener calificación del servicio', async () => {
    const res = await request(app)
      .get('/calificar-servicio/1'); // ID de chamba
    expect(res.statusCode).toBe(200);
    expect(res.body).toHaveProperty('calificacion');
  });
});
```

- Command: “/”
  NPM Test

TEST 2: Apache Benchmark (Basic Load):

- Screenshot:



- WSL Command: "localuser@DESKTOP-48S50AL:/mnt/c/Users/Benjamin_LZS/documents/Scalable System Design/TestProjects/chambaap/CHAMBAAP_grading$"
  ab -n 100 -c 10 http://localhost:3002/calificar-servicio

TEST 3: k6 (Stress):

- Screenshot:

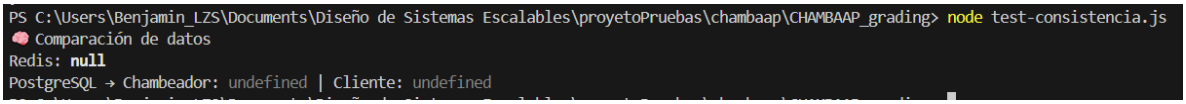
- Code: "/backend/auth-test.js"

```
import http from 'k6/http';
import { check } from 'k6';

export const options = {
  vus: 50,
  duration: '10s',
};

export default function () {
  const res = http.get('http://localhost:3002/calificar-servicio/1');
  check(res, {
    'status is 200': (r) => r.status === 200,
  });
}
```

- Command: “/backend”
  k6 run grating-test.js

TEST 4: Redis Consistency vs PostgreSQL

- Screenshot:



- Code:

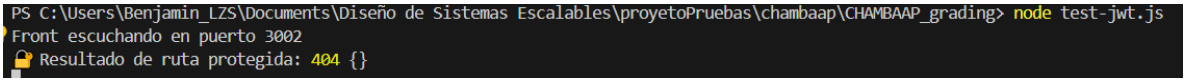
```
const { createClient } = require('redis');
const { Client } = require('pg');
const redis = createClient({ url: 'redis://localhost:6379' });

redis.connect().then(async () => {
  const pg = new Client({
    host: 'localhost',
    port: 5432,
    user: 'postgres',
    password: 'example',
    database: 'chambaap',
  });

  await pg.connect();

  const chamba_id = 1;
  const redisValor = await redis.get(`promedio:${chamba_id}`);
  const query = `
    SELECT ScoreForChambeador, ScoreForClient
    FROM Chamba
    WHERE Chamba_ID = $1
  `;
  const result = await pg.query(query, [chamba_id]);
  const { ScoreForChambeador, ScoreForClient } = result.rows[0] || {};

  console.log("🧠 Comparación de datos");
  console.log("Redis:", redisValor);
  console.log("PostgreSQL → Chambeador:", ScoreForChambeador, "| Cliente:",
ScoreForClient);

  await redis.quit();
  await pg.end();
```

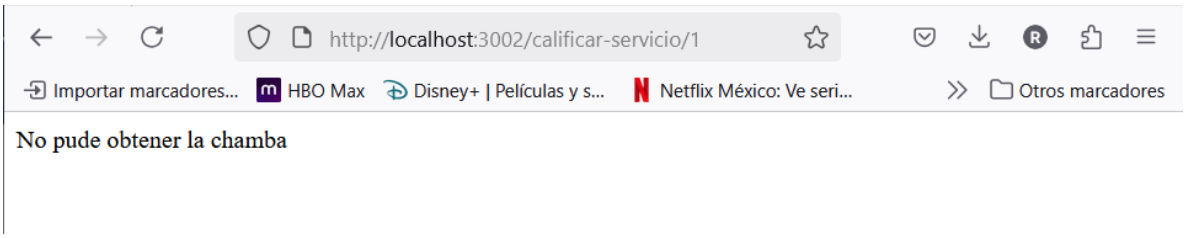
```
}).catch(console.error);
```

- Command: “/”
  Node test-consistencia.js

TEST 5: JWT and Route Protection

- Screenshot:





- Code:

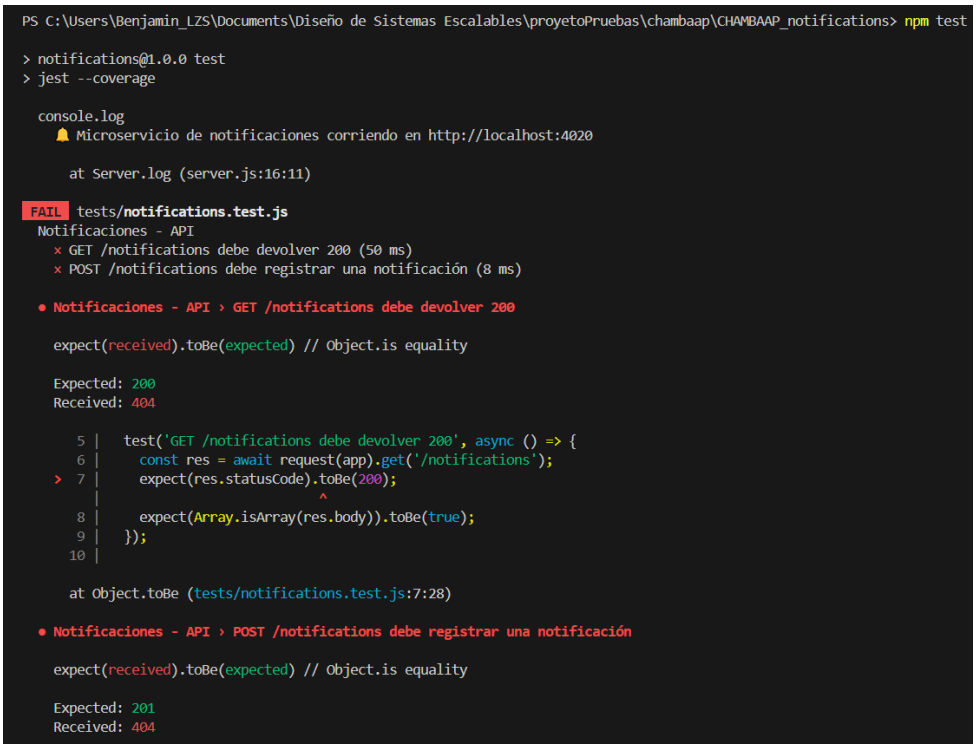
```
const request = require('supertest');
const app = require('./server');
const jwt = require('jsonwebtoken');

const token = jwt.sign(
  { user_id: 1, role: 'Client' },
  '.::lmNnjbbgTtdxxdEEWWZ<>ZsDr$%666yuijo90iJHvyg75$dd4f5',
  { expiresIn: '1h' }
);

request(app)
  .get('/private')
  .set('Authorization', `Bearer ${token}`)
  .end((err, res) => {
    if (err) throw err;
    console.log('🔐 Resultado de ruta protegida:', res.statusCode, res.body);
  });
```

- Command: “/”
  node test-jwt.js

CHAMBAAP_notifications

TEST 1: Jest + Supertest (Unitary / Integration / Coverage):

- Screenshot:



```
PS C:\Users\Benjamin_LZS\Documents\Diseño de Sistemas Escalables\proyetoPruebas\chambaap\CHAMBAAP_notifications> npm test

> notifications@1.0.0 test
> jest --coverage

  console.log
    🔔 Microservicio de notificaciones corriendo en http://localhost:4020

      at Server.log (server.js:16:11)

 FAIL  tests/notifications.test.js
  Notificaciones - API
    × GET /notifications debe devolver 200 (50 ms)
    × POST /notificaciones debe registrar una notificación (8 ms)

  ● Notificaciones - API › GET /notifications debe devolver 200

    expect(received).toBe(expected) // Object.is equality

    Expected: 200
    Received: 404

       5 |   test('GET /notificaciones debe devolver 200', async () => {
       6 |     const res = await request(app).get('/notifications');
     > 7 |     expect(res.statusCode).toBe(200);
         |                            ^
       8 |     expect(Array.isArray(res.body)).toBe(true);
       9 |   });
      10 |

      at Object.toBe (tests/notifications.test.js:7:28)

  ● Notificaciones - API › POST /notificaciones debe registrar una notificación

    expect(received).toBe(expected) // Object.is equality

    Expected: 201
    Received: 404
```

```
Expected: 200
Received: 404

   5 |   test('GET /notificaciones debe devolver 200', async () => {
   6 |     const res = await request(app).get('/notifications');
 > 7 |     expect(res.statusCode).toBe(200);
     |                            ^
   8 |     expect(Array.isArray(res.body)).toBe(true);
   9 |   });
  10 |

  at Object.toBe (tests/notifications.test.js:7:28)

● Notificaciones - API › POST /notificaciones debe registrar una notificación

expect(received).toBe(expected) // Object.is equality

Expected: 201
Received: 404

  18 |       });
  19 |
> 20 |     expect(res.statusCode).toBe(201);
     |                            ^
  21 |     expect(res.body).toHaveProperty('message');
  22 |   });
  23 | });

  at Object.toBe (tests/notifications.test.js:20:28)
```

```
      at Object.toBe (tests/notifications.test.js:19:28)

-----------------------------------|---------|----------|---------|---------|-------------------
File                               | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------------------------------|---------|----------|---------|---------|-------------------
All files                          |   74.07 |       25 |   57.14 |   76.47 |
 CHAMBAAP_notifications            |     100 |       50 |     100 |     100 |
  server.js                        |     100 |       50 |     100 |     100 | 14
 CHAMBAAP_notifications/controllers |   42.85 |        0 |     100 |   42.85 |
  notificationsController.js       |   42.85 |        0 |     100 |   42.85 | 7-18
 CHAMBAAP_notifications/models     |   66.66 |      100 |       0 |   66.66 |
  db.js                            |     100 |      100 |     100 |     100 |
  notificationsModel.js            |      50 |      100 |       0 |      50 | 4-14
 CHAMBAAP_notifications/routes     |     100 |      100 |     100 |     100 |
  notificationsRoutes.js           |     100 |      100 |     100 |     100 |
 CHAMBAAP_notifications/utils      |   76.92 |      100 |      50 |      90 |
  redis.js                         |   76.92 |      100 |      50 |      90 | 15
-----------------------------------|---------|----------|---------|---------|-------------------
Test Suites: 1 failed, 1 total
Tests:       2 failed, 2 total
Snapshots:   0 total
Time:        6.582 s, estimated 7 s
Ran all test suites.
Jest did not exit one second after the test run has completed.

'This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with `--detectOpenHandles` to troubleshoot this issue.
```

```
      at Object.toBe (tests/notifications.test.js:20:28)

 -----------------------------------|---------|----------|---------|---------|-------------------
 File                               | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
 -----------------------------------|---------|----------|---------|---------|-------------------
 All files                          |   68.51 |       25 |   28.57 |   72.54 |
  CHAMBAAP_notifications            |     100 |       50 |     100 |     100 |
   server.js                        |     100 |       50 |     100 |     100 | 14
  CHAMBAAP_notifications/controllers |   28.57 |        0 |       0 |   28.57 |
   notificationsController.js       |   28.57 |        0 |       0 |   28.57 | 5-18
  CHAMBAAP_notifications/models     |   66.66 |      100 |       0 |   66.66 |
   db.js                            |     100 |      100 |     100 |     100 |
   notificationsModel.js            |      50 |      100 |       0 |      50 | 4-14
  CHAMBAAP_notifications/routes     |     100 |      100 |     100 |     100 |
   notificationsRoutes.js           |     100 |      100 |     100 |     100 |
  CHAMBAAP_notifications/utils      |   69.23 |      100 |      25 |      90 |
   redis.js                         |   69.23 |      100 |      25 |      90 | 15
 -----------------------------------|---------|----------|---------|---------|-------------------
 Test Suites: 1 failed, 1 total
 Tests:       2 failed, 2 total
 Snapshots:   0 total
 Time:        1.681 s, estimated 2 s
 Ran all test suites.
 Jest did not exit one second after the test run has completed.

 'This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with `--detectOpenHandles` to troubleshoot this issue.
```

```
'This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with `--detectOpenHandles` to troubleshoot this issue.
  console.error
    ✕ Redis Error: Error: getaddrinfo ENOTFOUND redis
        at GetAddrInfoReqWrap.onlookupall [as oncomplete] (node:dns:118:26) {
      errno: -3008,
      code: 'ENOTFOUND',
      syscall: 'getaddrinfo',
      hostname: 'redis'
    }



      at Commander.<anonymous> (utils/redis.js:311:18)
      at RedisSocket.<anonymous> (node_modules/@redis/client/dist/lib/client/index.js:422:14)
      at RedisSocket._RedisSocket_connect (node_modules/@redis/client/dist/lib/client/socket.js:166:18)
      at Commander.connect (node_modules/@redis/client/dist/lib/client/index.js:185:9)
      at utils/redis.js:322:5
```

- Code: "/tests/chat.test.js"

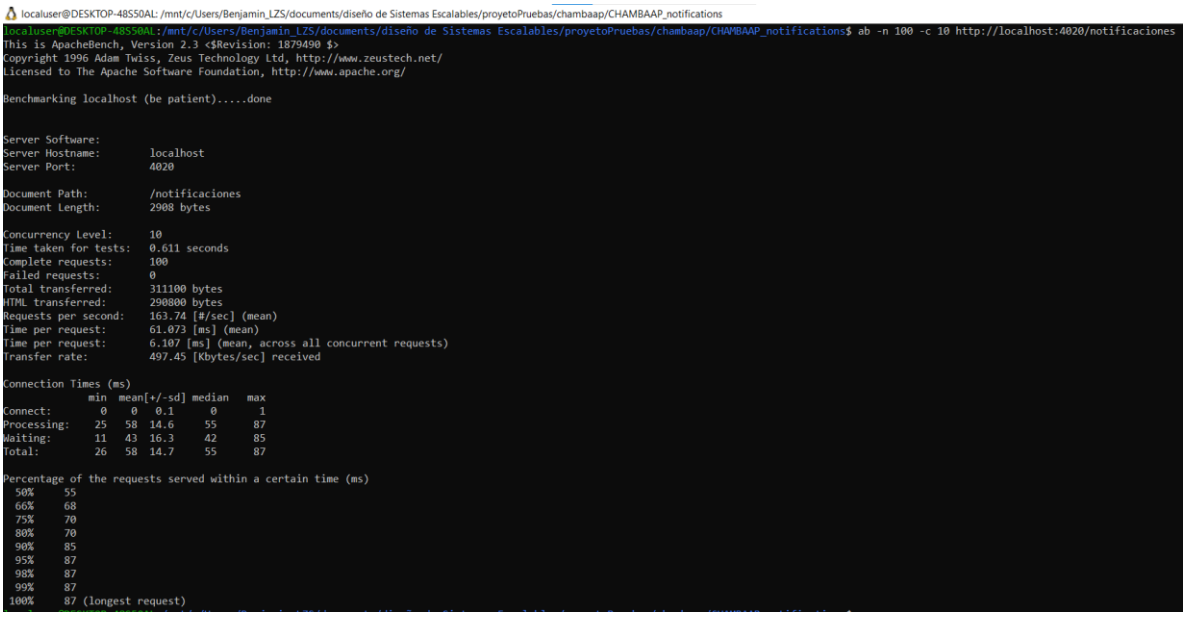
```
const request = require('supertest');
const app = require('../server');
describe('Notificaciones - API', () => {
 test('GET /notificaciones debe devolver 200', async () => {
   const res = await request(app).get('/notificaciones');
   expect(res.statusCode).toBe(200);
   expect(Array.isArray(res.body)).toBe(true);
 });

 test('POST /notifications debe registrar una notificación', async () => {
   const res = await request(app)
     .post('/notificaciones')
```

```
      .send({
        chamba_id: 1,
        client_id: 1,
        mensaje: "Notificación de prueba"
      });

    expect(res.statusCode).toBe(201);
    expect(res.body).toHaveProperty('message');
  });
});
```

- Command: “/”
  npm test

TEST 2: Apache Benchmark (Basic Load):

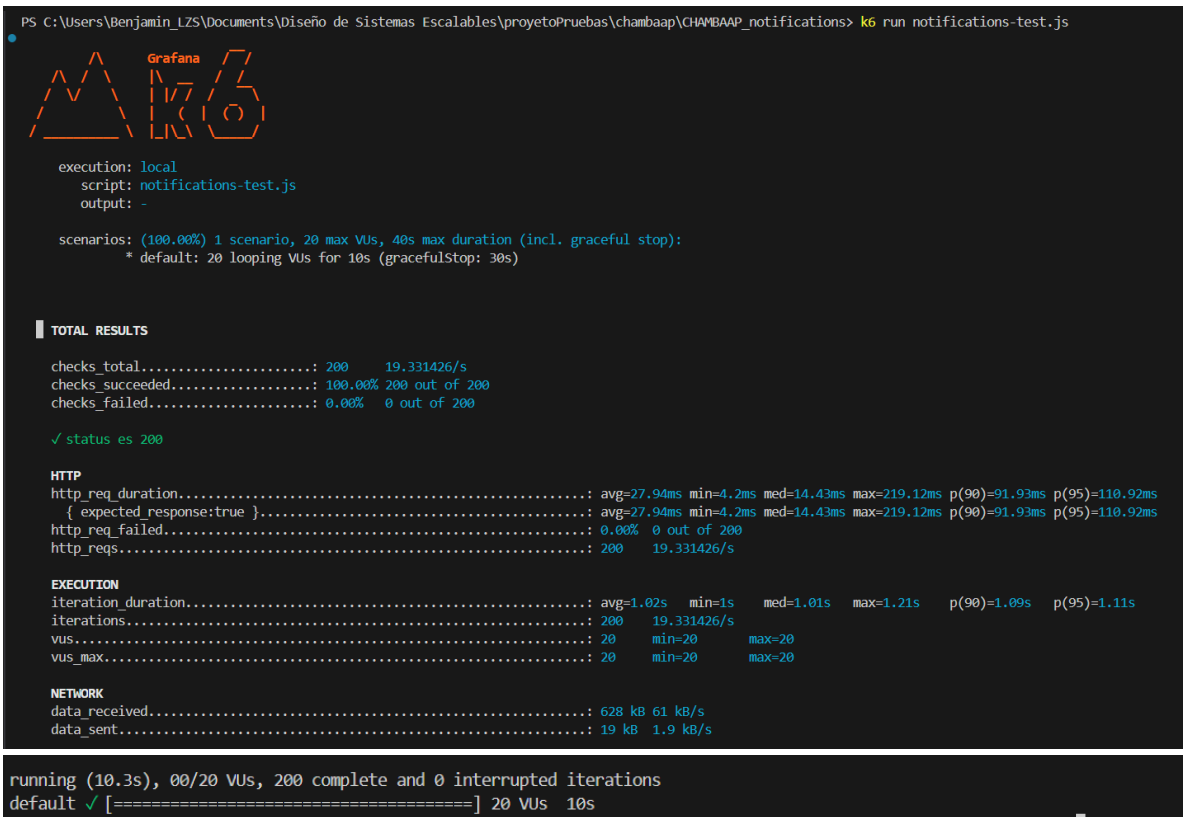
- Capture:



- WSL Command: "localuser@DESKTOP-48S50AL:/mnt/c/Users/Benjamin_LZS/documents/scalable system design/test//chambaap/CHAMBAAP_notifications$"
  ab -n 100 -c 10 http://localhost:4020/notificaciones

TEST 3: k6 (Stress):

- Screenshot:



```
PS C:\Users\Benjamin_LZS\Documents\Diseño de Sistemas Escalables\proyetoPruebas\chambaap\CHAMBAAP_notifications> k6 run notifications-test.js

     /\      Grafana   /‾‾/
    /\  /  \     |\  __   /  /
   /  \/    \    | |/ /  /   ‾‾\
  /          \   |   (  |  (‾)  |
 / __________ \  |_|\_\  \_____/

     execution: local
        script: notifications-test.js
        output: -

     scenarios: (100.00%) 1 scenario, 20 max VUs, 40s max duration (incl. graceful stop):
              * default: 20 looping VUs for 10s (gracefulStop: 30s)


  TOTAL RESULTS

    checks_total.....................: 200     19.331426/s
    checks_succeeded.................: 100.00% 200 out of 200
    checks_failed....................: 0.00%   0 out of 200

    ✓ status es 200

    HTTP
    http_req_duration..............................................: avg=27.94ms min=4.2ms med=14.43ms max=219.12ms p(90)=91.93ms p(95)=110.92ms
      { expected_response:true }...................................: avg=27.94ms min=4.2ms med=14.43ms max=219.12ms p(90)=91.93ms p(95)=110.92ms
    http_req_failed................................................: 0.00%  0 out of 200
    http_reqs......................................................: 200    19.331426/s

    EXECUTION
    iteration_duration.............................................: avg=1.02s   min=1s    med=1.01s   max=1.21s    p(90)=1.09s   p(95)=1.11s
    iterations.....................................................: 200    19.331426/s
    vus............................................................: 20     min=20       max=20
    vus_max........................................................: 20     min=20       max=20

    NETWORK
    data_received..................................................: 628 kB 61 kB/s
    data_sent......................................................: 19 kB  1.9 kB/s

running (10.3s), 00/20 VUs, 200 complete and 0 interrupted iterations
default ✓ [======================================] 20 VUs  10s
```

- Code: "/backend/auth-test.js"

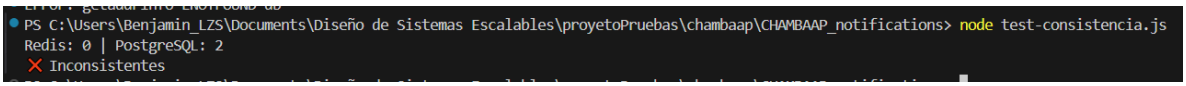
```
import http from 'k6/http';
import { sleep, check } from 'k6';

export let options = {
  vus: 20,
  duration: '10s',
};

export default function () {
  let res = http.get('http://localhost:4020/notificaciones');
  check(res, {
    'status es 200': (r) => r.status === 200,
  });
  sleep(1);
}
```

- Command: “/”

k6 run notofications-test.js

TEST 4: Redis Consistency vs PostgreSQL

- Screenshot:



- SQL Code: "PostgreSQL"

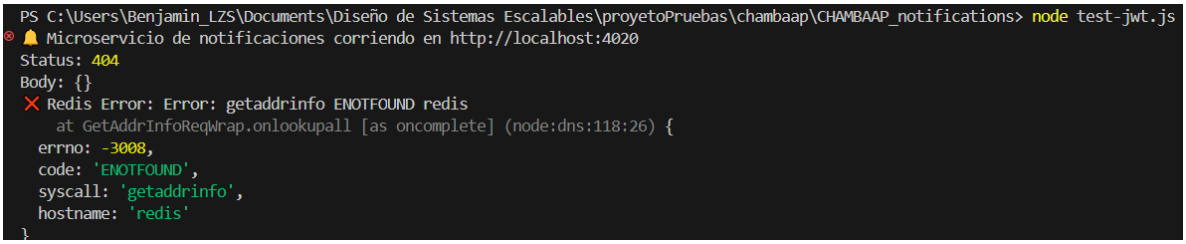
```
const { createClient } = require('redis');
const { Client } = require('pg');

const redisClient = createClient();
const pgClient = new Client({
  host: 'localhost',
  user: 'postgres',
  password: 'example',
  database: 'chambaap',
  port: 5432,
});

async function verificarConsistencia() {
  try {
    await redisClient.connect();
    await pgClient.connect();

    const redisKeys = await redisClient.keys('notificaciones:*');
    const redisCount = redisKeys.length;

    const res = await pgClient.query('SELECT COUNT(*) FROM Notifications');
    const pgCount = parseInt(res.rows[0].count);

    console.log(`Redis: ${redisCount} | PostgreSQL: ${pgCount}`);

    if (redisCount === pgCount) {
      console.log('✅ Consistentes');
    } else {
      console.log('❌ Inconsistentes');
    }
  } catch (err) {
    console.error('Error:', err.message);
  } finally {
    await redisClient.disconnect();
    await pgClient.end();
  }
}
```

```
verificarConsistencia();
```

- Commando: “/”
  Node test-consistencia.js

EXHIBIT 5: JWT and Route Protection

- Screenshot:



- Code:

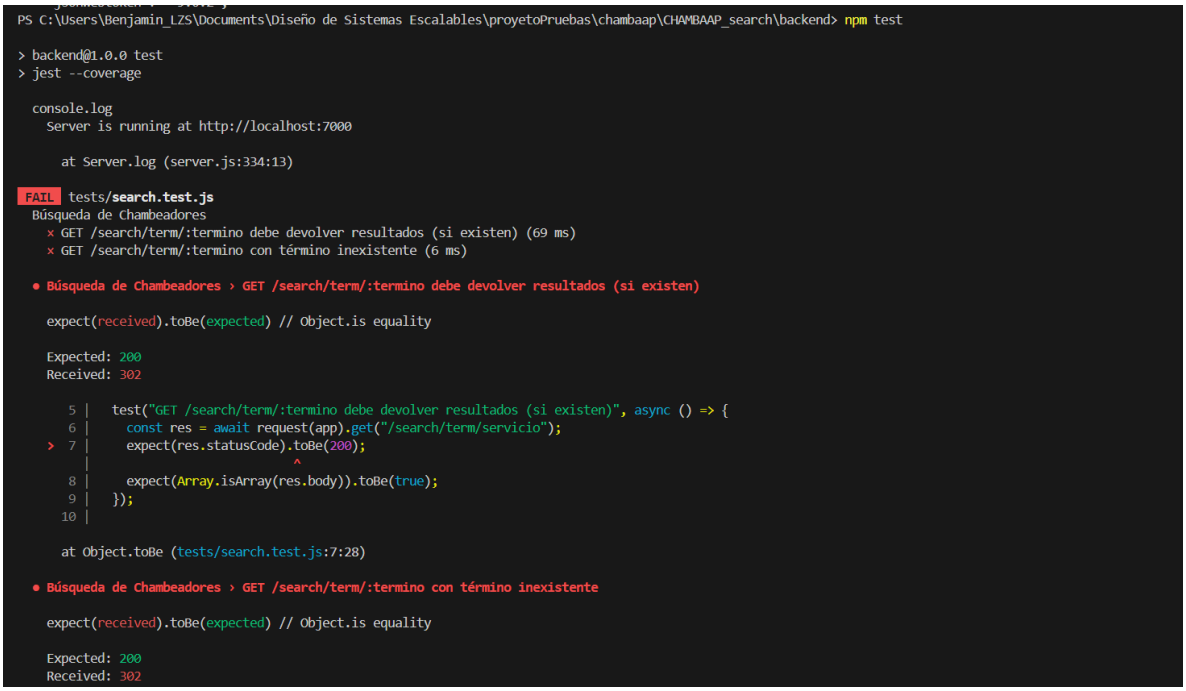
```
const request = require('supertest');
const app = require('./server');

const fakeToken = 'Bearer faketoken.123';

request(app)
  .get('/private')
  .set('Authorization', fakeToken)
  .then(res => {
    console.log('Status:', res.statusCode);
    console.log('Body:', res.body);
  })
  .catch(console.error);
```

- Command: "/"
  node test-jwt.js

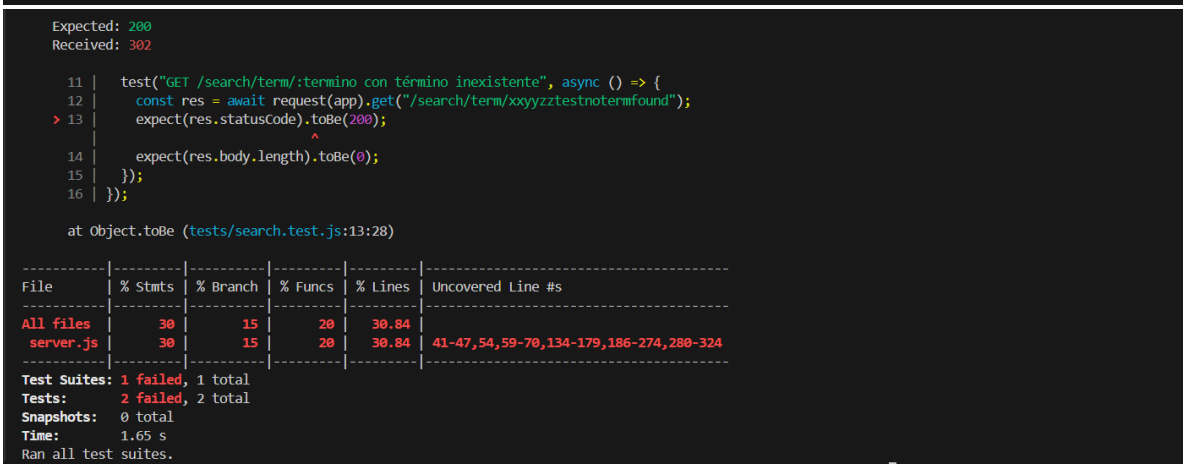
CHAMBAAP_search

TEST 1: Jest + Supertest (Unitary / Integration / Coverage):

- Screenshot:



```
PS C:\Users\Benjamin_LZS\Documents\Diseño de Sistemas Escalables\proyetoPruebas\chambaap\CHAMBAAP_search\backend> npm test

> backend@1.0.0 test
> jest --coverage

  console.log
    Server is running at http://localhost:7000

      at Server.log (server.js:334:13)

 FAIL  tests/search.test.js
  Búsqueda de Chambeadores
    × GET /search/term/:termino debe devolver resultados (si existen) (69 ms)
    × GET /search/term/:termino con término inexistente (6 ms)

  ● Búsqueda de Chambeadores › GET /search/term/:termino debe devolver resultados (si existen)

    expect(received).toBe(expected) // Object.is equality

    Expected: 200
    Received: 302

       5 |   test("GET /search/term/:termino debe devolver resultados (si existen)", async () => {
       6 |     const res = await request(app).get("/search/term/servicio");
     > 7 |     expect(res.statusCode).toBe(200);
         |                            ^
       8 |     expect(Array.isArray(res.body)).toBe(true);
       9 |   });
      10 |

      at Object.toBe (tests/search.test.js:7:28)

  ● Búsqueda de Chambeadores › GET /search/term/:termino con término inexistente

    expect(received).toBe(expected) // Object.is equality

    Expected: 200
    Received: 302
```



```
    Expected: 200
    Received: 302

      11 |   test("GET /search/term/:termino con término inexistente", async () => {
      12 |     const res = await request(app).get("/search/term/xxyyzztestnotermfound");
    > 13 |     expect(res.statusCode).toBe(200);
         |                            ^
      14 |     expect(res.body.length).toBe(0);
      15 |   });
      16 | });

      at Object.toBe (tests/search.test.js:13:28)

-----------|---------|----------|---------|---------|-------------------------------------
File       | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------|---------|----------|---------|---------|-------------------------------------
All files  |      30 |       15 |      20 |   30.84 |
 server.js |      30 |       15 |      20 |   30.84 | 41-47,54,59-70,134-179,186-274,280-324
-----------|---------|----------|---------|---------|-------------------------------------
Test Suites: 1 failed, 1 total
Tests:       2 failed, 2 total
Snapshots:   0 total
Time:        1.65 s
Ran all test suites.
```

- Code: "/tests/chat.test.js"

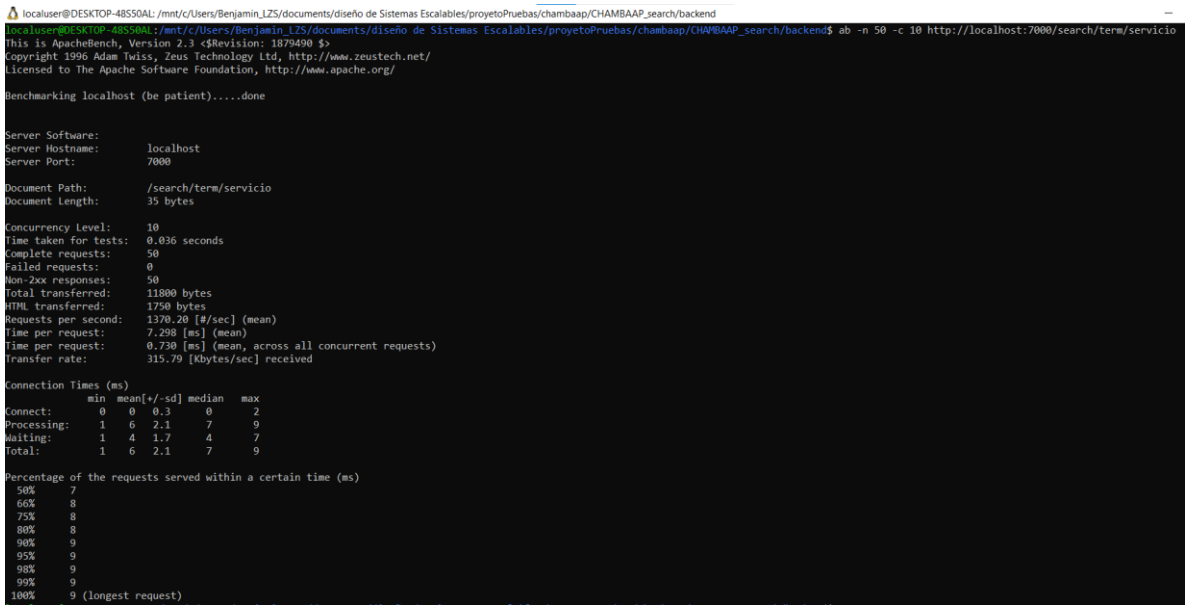
```
const request = require('supertest');
const app = require('../server');

describe("Búsqueda de Chambeadores", () => {
  test("GET /search/term/:termino debe devolver resultados (si existen)", async ()
=> {
    const res = await request(app).get("/search/term/servicio");
    expect(res.statusCode).toBe(200);
```

```
    expect(Array.isArray(res.body)).toBe(true);
  });

  test("GET /search/term/:termino con término inexistente", async () => {
    const res = await request(app).get("/search/term/xxyyzztestnotermfound");
    expect(res.statusCode).toBe(200);
    expect(res.body.length).toBe(0);
  });
});
```

- Command: “/backend”
  npm test

TEST 2: Apache Benchmark (Basic Load):

- Screenshot:

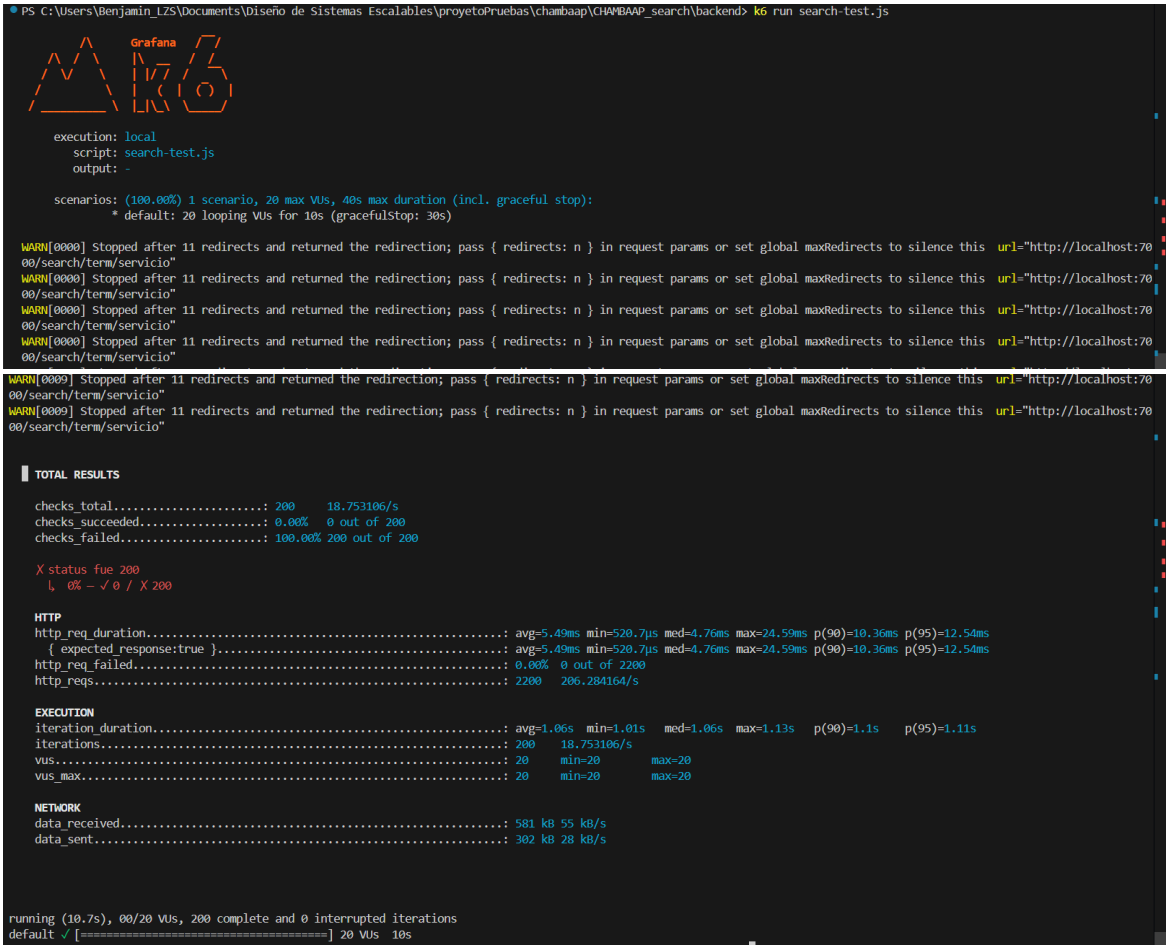


- Command WSL: “localuser@DESKTOP-48S50AL:/mnt/c/Users/Benjamin_LZS/documents/Scalable System Design/TestProjects/ chambaap/CHAMBAAP_search/backend$"

ab -n 50 -c 10 http://localhost:7000/search/term/servicio

TEST 3: k6 (Stress):

- Screenshot:



- Code: "/backend/search-test.js"

```
import http from 'k6/http';
import { check, sleep } from 'k6';
export let options = {
  vus: 20,
  duration: '10s',
};
export default function () {
  let res = http.get('http://localhost:7000/search/term/servicio');
  check(res, { 'status fue 200': (r) => r.status === 200 });
  sleep(1);
}
```

- Command: “/backend”

k6 run search-test.js

Bibliografias

- IEEE. (2008). IEEE Standard for Software Test Documentation (IEEE 829-2008). https://ieeexplore.ieee.org/document/4601589
- Jest. (2024). Jest Documentation. https://jestjs.io/docs/getting-started
- Postman. (2024). Postman API Platform. https://www.postman.com/
- Node.js Foundation. (2024). Node.js Documentation. https://nodejs.org/en/docs
- Express.js. (2024). Express Web Framework (Node.js). https://expressjs.com/
- PostgreSQL Global Development Group. (2024). PostgreSQL 16 Documentation. https://www.postgresql.org/docs/
- Redis. (2024). Redis Documentation. https://redis.io/docs/
- Apache Software Foundation. (2024). Apache Benchmark Tool (ab). https://httpd.apache.org/docs/2.4/programs/ab.html
- Grafana Labs. (2024). k6 Load Testing Tool. https://k6.io/docs/
- Locust. (2024). Locust Load Testing. https://docs.locust.io/
- OWASP. (2024). OWASP JWT Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html
- Mozilla Developer Network. (2024). HTTP Status Codes. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
- GitHub. (2024). Supertest GitHub Repository. https://github.com/visionmedia/supertest
- Docker. (2024). Docker Documentation. https://docs.docker.com/

Fin del Documento...

continuara