Universidad de Guadalajara
Centro Universitario de los Valles

Web system for congress management

Master's Degree in Software Engineering

Professor: Dr. Omar Ali Zatarain Durán
Author: Gutiérrez Constantino Roberto Carlos

1. Introduction

This document presents all the activities and artifacts developed during the Software Configuration Management process for the WebCongress project, carried out between January and May 2025. WebCongress is a system designed to manage registration, access control, and real-time interaction of attendees at academic and professional conferences.

This report summarizes the SCM practices applied, including configuration identification and control, auditing, and status logging. This final report references the various documents used in the implementation of the system and highlights the key actions and decisions taken throughout the process.

The SCM process was crucial in ensuring traceability, change control, and version consistency throughout the system. By establishing baselines, managing change requests (CR001 and CR002), and conducting audits, the team ensured the proper maintenance and validation of all project elements. This report provides a consolidated view of how SCM contributed to the stability and quality of the final product.

2. Software Configuration Management Process

The Software Configuration Management (SCM) process implemented for the WebCongress project was key to ensuring control, traceability, and orderly evolution of the software from February to May 2025. It consisted of four main activities: configuration identification, configuration control, status accounting, and configuration audits. Each task was carefully applied to adapt to the project's scope, time, and resource constraints.

2.1 Configuration Identification

This task focused on identifying and documenting all the Configuration Items (CIs) that formed the system. In WebCongress, CIs included:

- Source code files
- UI components for user registration and interaction
- Configuration files
- Project documentation: user manuals, test plans, installation guides
- Test scripts and test data files

Each CI was assigned a unique identifier and version label, then catalogued in the configuration registry. These items were grouped into baselines (e.g., baseline v1.2), which represented stable reference points from which further changes were controlled. This identification process ensured every key artifact in the project was traceable and well defined.

2.2 Configuration Control

This process ensured that any proposed change to the system was properly evaluated, approved, and implemented. During development, the team submitted and processed two major Change Requests (CR001 and CR002). CR001 introduced a networking module, and CR002 integrated a secure payment gateway.

Each change was analyzed by the Change Control Board (CCB), composed of the project manager, quality assurance lead, backend developer, and financial advisor. The analysis included technical feasibility, estimated effort, risks, and impact on schedule and budget. Once approved, changes were documented and implemented in accordance with the project's standards and updated in the baseline and CI registry.

2.3 Status Accounting

Status accounting consisted of continuously recording the current state of all configuration items and approved changes. For WebCongress, this involved tracking the status of CRs at three key milestones: project beginning, 50% development progress, and 90% completion.

For each CR, the team documented:

- Estimated vs. actual budget and time
- Risk level and outcome status (Solved, Addressing, Unsolved)
- Rules applied (e.g., Rule 1: if time exceeded, provide justification)
- Implementation classification (Planned, Slower, Faster)
- Feasibility of implementation given current HR and resources
- This process provided stakeholders with full visibility of project evolution and helped anticipate delays, over-budget risks, and technical gaps.

2.4 Configuration Audits

To verify the correctness and completeness of the final system, the team conducted both Functional Configuration Audit (FCA) and Physical Configuration Audit (PCA).

- The FCA verified that each requirement was implemented correctly and that all related test cases passed. This ensured that the system behaved as expected.
- The PCA checked that all deliverables (code, documentation, build files) matched the recorded configuration and were properly labeled and versioned.

Audit reports were generated, and any inconsistencies or missing artifacts were documented. Minor corrections were applied before delivery to ensure

full compliance. These audits confirmed that the system was in a stable and acceptable state for release.

3. History of Changes

The following entries summarize the most relevant configuration changes made during the WebCongress project. Each entry includes data related to the document, its version history, configuration items involved, and repository.

This section presents the chronological record of key documents and changes tracked during the WebCongress project lifecycle.

Table 3.1 Presentation of the first project baseline.

Date	2025-01-31	**Document**	Requirements Specification Document
Configuration items Desciption		Initial list of functional and non-functional requirements for WebCongress v1.0	
Document on the repository of the project:		Yes	
Previous document version:		N/A	
Comments:			
First baseline created; served as the basis for early design planning			

Table 3.2 Update for the project baseline

Date	2025-02-07	**Document**	CR001 – Networking Module Request
Configuration items Desciption		Proposed addition of attendee chat and real-time interaction functionality	
Document on the repository of the project:		Yes	
Previous document version:		N/A	
Comments:			
Submitted by frontend developer; impact analysis initiated.			

Table 3.3 First change request added.

Date	2025-02-21	**Document**	Approved CR001 & Updated Requirements
Configuration items Desciption		Chat module requirements integrated into v1.1 baseline	
Document on the repository of the project:		Yes	
Previous document version:		v1.0	
Comments:			
Updated after approval by the Change Control Board.			

Table 3.4 Addition of 3 change requests.

Date	2025-03-07	**Document**	Payment Integration Request
Configuration items Desciption		Request to integrate Stripe and PayPal payment gateways	
Document on the repository of the project:		Yes	
Previous document version:		N/A	
Comments:			
Submitted due to stakeholder requirement for online registration payments.			

Table 3.5 Update of design document after change approvals.

Date	2025-03-21	**Document**	Approved CR002 & Updated Design Document
Configuration items Desciption		Payment logic and UI modifications included in system design	
Document on the repository of the project:		Yes	
Previous document version:		v1.1	
Comments:			
Included CI updates for backend controller and frontend form			

Table 3.6 Audit verification process completed prior to reease

Date	2025-04-11	**Document**	onfiguration Audit Report (FCA + PCA)
Configuration items Desciption		Full review of system files, documents, and test cases	
Document on the repository of the project:		Yes	
Previous document version:		N/A	
Comments:			
Minor discrepancies corrected before release.			

Table 3.7 Final version deployed

Date	2025-05-02	**Document**	Final Release Package v1.2
Configuration items Desciption		Complete WebCongress system with verified components	
Document on the repository of the project:		No	
Previous document version:		Internal test build	
Comments:			
Approved for delivery and archived			

4. Configuration Control

The configuration control process in WebCongress was essential to ensure that all proposed changes were properly reviewed, authorized, documented, and implemented without compromising project quality or stability.

4.1 Objectives

- Establish a standard method to evaluate and process change requests (CRs)
- Ensure only approved changes are applied to baselines and configuration items
- Maintain traceability and documentation of each decision

4.2 Change Request Management

Throughout the project, the development team submitted multiple Change Requests. Two major ones were processed:

- **CR001**: Addition of a real-time networking module
- **CR002**: Integration of Stripe and PayPal for payment functionality

Each CR included:

- Description of the proposed change
- Justification (technical, functional, or external compliance)
- Estimated time, effort, and budget impact
- Associated configuration items and baseline version

4.3 Change Control Board (CCB)

All change requests were reviewed by the Change Control Board (CCB), which included:

- Project Manager
- Configuration Manager
- QA Lead
- Backend Developer
- Financial Advisor

The CCB evaluated each CR according to:

- Urgency and necessity
- Alignment with system goals
- Impact on schedule and cost
- Team feasibility and resources

Each CR was either approved, rejected, or deferred, and the decision was documented with justification.

4.4 Change Implementation

Once a CR was approved:

- A developer was assigned to implement the change
- The affected configuration items were updated
- Tests were conducted (unit, integration, regression)
- Documentation was updated accordingly
- Status accounting and audit logs were revised

4.5 Traceability and Records

Each approved change was tracked from submission to implementation. Documentation included:

- CR form and approval date
- Baseline affected
- CIs modified and versioned
- Test results and audit checklists
- Comments and final status (Solved, Addressing, Unsolved)

This control ensured that WebCongress maintained a consistent, reliable, and traceable configuration throughout its development, especially when facing external pressures and evolving functional requirements.

Universidad de Guadalajara

Centro Universitario de los Valles

Web system for congress management

SCM-SIPaF - V1.1

Master's Degree in Software Engineering

Professor: Dr. Omar Ali Zatarain Durán

Author: Gutiérrez Constantino Roberto Carlos

Change control

Revision	Description	Author	Date	Version
0.1	Preliminary version	Roberto Gutiérrez	02/15/2025	SCM-WebCongress-V0.1
0.2	Requirements, architecture and SCM sections added	Roberto Gutiérrez	03/01/2025	SCM-WebCongress-V0.2
1.0	CR001 (Networking) and CR002 (Payments) applied	Roberto Gutiérrez	04/20/2025	SCM-WebCongress-V1.0
1.1	Adjustments after FCA/PCA review	Roberto Gutiérrez	05/05/2025	SCM-WebCongress-V1.1

Revision

Version	Responsible	Date	Status
1.0	Dr. Omar Zatarain	05/19/2025	All CIs updated and verified.

1. Introduction

To support the organization and automation of academic and professional congresses, especially for participants across institutions, the WebCongress project was developed between February and May 2025. This system allows the creation and management of digital congresses through a platform that includes modules for visitor access, attendee registration and payment, and administrative control of events.

Traditional event registration and control systems often rely on spreadsheets, emails, and external tools that complicate real-time monitoring, limit interaction between attendees, and introduce human errors. WebCongress addresses these problems by centralizing all critical features into a web-based platform that is accessible, reliable, and secure.

The platform works as follows:

1. The admin registers a congress and its details.
2. Attendees register through a public page, select their congress, and pay securely.
3. Real-time modules like chat or networking are activated during the event.
4. Administrators can monitor attendance, validate payments, and export reports.

1.1 Purpose

The purpose of this document is to define the functional and non-functional specifications of the WebCongress system and describe the configuration management process used during its development. It presents the design, change request management, version control, and auditing practices applied.

1.2 Scope of the system

The WebCongress system enables congress managers to publish their events, register participants, receive payments, and activate real-time features during the conference. It includes:

- Visitor interface (read-only access to event information)
- Attendee module (registration, login, payment)
- Admin dashboard (event creation, monitoring, export of data)

It supports event customization, secure payment handling (via Stripe and PayPal), and audit tracking of every configuration item.

1.3 Acronyms details

Term	Definition
SCM	Software Configuration Management
CR	Change Request
CI	Configuration Item
FCA	Functional Configuration Audit
PCA	Physical Configuration Audit
PM	Project Manager
QA	Quality Assurance
DB	Database
UI	User Interface

1.4 Document overview

This document provides a comprehensive overview of the software configuration management process followed during the development of the WebCongress project. It includes the context and motivation behind the system, the methodology used to identify and control configuration items, the handling of change requests, and the application of audits and traceability techniques to ensure system integrity and proper delivery.

2. General Description

This section provides a general overview of the WebCongress system, its operating environment, users, and design constraints.

2.1 Product Perspective

WebCongress is a web-based system designed to support the organization, administration, and execution of digital congresses. It works as an independent platform but can also be integrated into broader university or institutional portals. It follows a modular architecture, with components such as user registration, payment processing, administrative dashboard, and real-time attendee interaction.

The system includes:

- A public portal for visitors to view event information.
- A registration and login system for attendees.
- Payment processing through third-party gateways (Stripe, PayPal).
- A backend dashboard for event management and report generation.

2.2 Product Functions

Key functionalities of WebCongress include:

- Event creation and customization by admins.
- User registration and profile management.
- Online payment processing and validation.
- Attendance tracking and reporting.
- Chat or networking functionality during live events.

2.3 User Characteristics

WebCongress is designed for three main types of users:

- **Visitors**: Can access general event information without registration.
- **Attendees**: Can register, log in, purchase access, and participate in event features.
- **Administrators**: Can create, monitor, and manage events, view analytics, and manage attendee records.

No advanced technical knowledge is required from visitors or attendees. Administrators should have basic familiarity with web platforms and data management.

2.4 Constraints

The project was developed under the following constraints:

- Fixed delivery deadline: may 2025.
- Budget capped and 80% allocated by mid-development.
- Team availability fully committed; no extra resources.
- Compliance with external standards such as IMV codes for surgical items (simulated scenario).

2.5 Assumptions and Dependencies

- Users will have internet access and modern browsers (Chrome, Firefox).
- External services (Stripe, PayPal) are available and stable.
- Institutional support for event publication and outreach will be provided.
- The platform is hosted on a secure server with periodic backups.

This general description establishes the foundation for the system's design, management, and configuration activities.

3. Specific Requirements

This section presents the specific functional and non-functional requirements defined for the WebCongress system. These requirements were gathered through stakeholder interviews, analysis of similar platforms, and the expected functionalities required for managing academic congresses.

3.1 Functional Requirements

The functional requirements describe what the system must do to meet its goals. The WebCongress system must provide features for:

- User registration, login, and authentication
- Purchasing passes (free and paid) for events
- Registering for workshops or conferences, depending on seat availability
- Viewing detailed information about workshops and speakers
- Creating, editing, and deleting congresses and their sessions
- Managing speakers and event packages
- Assigning activities and managing attendees
- Displaying a navigation menu with logical grouping of sections
- Filtering and consulting events by congress

• ***Code***	***Name***		***Priority level***
RFU-001	User registration and authentication		High
Description	The system must support the following: user registration and authentication.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-002	Purchase of congress passes		High
Description	The system must support the following: purchase of congress passes (free and complete).		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-003	Registration for workshops or conferences, subject to availability		High
Description	The system must support the following: registration for workshops or conferences, subject to availability.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		
Code	***Name***		***Priority level***
RFU-004	View information on workshops and conferences (name, description, time, place, speaker)		High
Description	The system must support the following: view information on workshops and conferences (name, description, time, place, speaker).		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency

Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.

Code	***Name***		***Priority level***
RFU-005	View available packages		High
Description	The system must support the following: view available packages.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-005	View available packages		High
Description	The system must support the following: view available packages.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-006	Speaker management: registration, editing, deletion		High
Description	The system must support the following: speaker management: registration, editing, deletion		
Inputs	**Source**	**Outputs**	**Constraints**

User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-007	Event management: registration, editing, deletion of workshops and conferences		High
Description	The system must support the following: event management: registration, editing, deletion of workshops and conferences		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-008	View registered users, including emails and purchased packages		High
Description	The system must support the following: view registered users, including emails and purchased packages.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-009	Top menu with links to Event, Packages, Workshops/Conferences, and Buy Pass		High
Description	The system must support the following: top menu with links to event, packages, workshops/conferences, and buy pass.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-010	Creation of new congresses with relevant data		High
Description	The system must support the following: top menu with links to event, packages, workshops/conferences, and buy pass.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-011	Editing and deleting existing congresses		High
Description	The system must support the following: editing and deleting existing congresses.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency

Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.

Code	***Name***		***Priority level***
RFU-012	Assignment of workshops, conferences, and speakers to congresses		High
Description	The system must support the following: assignment of workshops, conferences, and speakers to congresses.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-013	Consultation of events grouped by congress		High
Description	The system must support the following: consultation of events grouped by congress .		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RFU-014	Selection of congress when registering or buying passes		High
Description	The system must support the following: selection of congress when registering or buying passes.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

3.2 Non-Functional Requirements

The non-functional requirements define quality attributes and system constraints that ensure usability, performance, and reliability. WebCongress must:

- Offer an intuitive and user-friendly interface
- Be compatible with modern web browsers (Chrome, Firefox)
- Support concurrent users accessing the system simultaneously
- Encrypt user data and sensitive information
- Protect against common attacks such as SQL injection
- Remain available 24/7 without interruption
- Provide a clear interface for managing multiple congresses
- Organize data efficiently for reporting and analysis

Code	***Name***		***Priority level***
RNF-001	Intuitive and user-friendly design		High
Description	The system must support the following: intuitive and user-friendly design.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RNF-002	Compatibility with modern browsers		High
Description	The system must support the following: compatibility with modern browsers.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RNF-003	Ability to manage multiple simultaneous users		High
Description	The system must support the following: ability to manage multiple simultaneous users.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		
Code	***Name***		***Priority level***

RNF-004	Encryption of passwords and sensitive data		High
Description	The system must support the following: encryption of passwords and sensitive data.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required (e.g., name, email, selection, filters)	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RNF-005	Prevention of attacks such as SQL injection		High
Description	The system must support the following: prevention of attacks such as sql injection		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RNF-006	24/7 system availability		High
Description	The system must support the following: 24/7 system availability.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		
Code	***Name***		***Priority level***

RNF-007	Clear interface for congress selection and management		High
Description	The system must support the following: clear interface for congress selection and management.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

Code	***Name***		***Priority level***
RNF-008	Efficient data organization for multiple events		High
Description	The system must support the following: efficient data organization for multiple events.		
Inputs	**Source**	**Outputs**	**Constraints**
User data as required	User or Administrator	Confirmation message, updated display, or relevant result	Must validate all inputs and prevent duplication or inconsistency
Process	1. The user accesses the corresponding module. 2. The system displays the input form or options. 3. The user submits information, and the system processes the request.		

3.3 Traceability

Each requirement is traceable to its corresponding configuration items (CIs), change requests (CRs), and baseline versions. This allows tracking of its implementation, testing, and updates throughout the project lifecycle.

These specific requirements served as the foundation for design, testing, configuration control, and auditing activities conducted throughout the project.

4. Design

This section outlines the design approach followed during the development of WebCongress, including architectural structure, component modules, and interface considerations.

4.1 Architectural Design

WebCongress follows a modular, client-server architecture. The backend is developed using Python with Flask, while the frontend is built with HTML, CSS, and JavaScript (with optional integration of React for dynamic content). The system relies on RESTful APIs to communicate between modules.

- **Frontend:** User interfaces for visitors, attendees, and administrators
- **Backend:** Business logic for user authentication, event management, and payment processing
- **Database:** Structured storage using PostgreSQL or MySQL, depending on deployment
- **External Services:** Integration with Stripe and PayPal for payment processing

All components are containerized using Docker to ensure environment consistency.

4.2 Component Design

The system is divided into three main modules:

- **Visitor Module:** Allows general public users to browse information without authentication.
- **Attendee Module:** Handles user registration, login, profile management, and purchase of passes.
- **Admin Module:** Allows creation and management of events, assignment of speakers, and real-time monitoring of event activity.

Each module is further subdivided into controllers, services, and views.

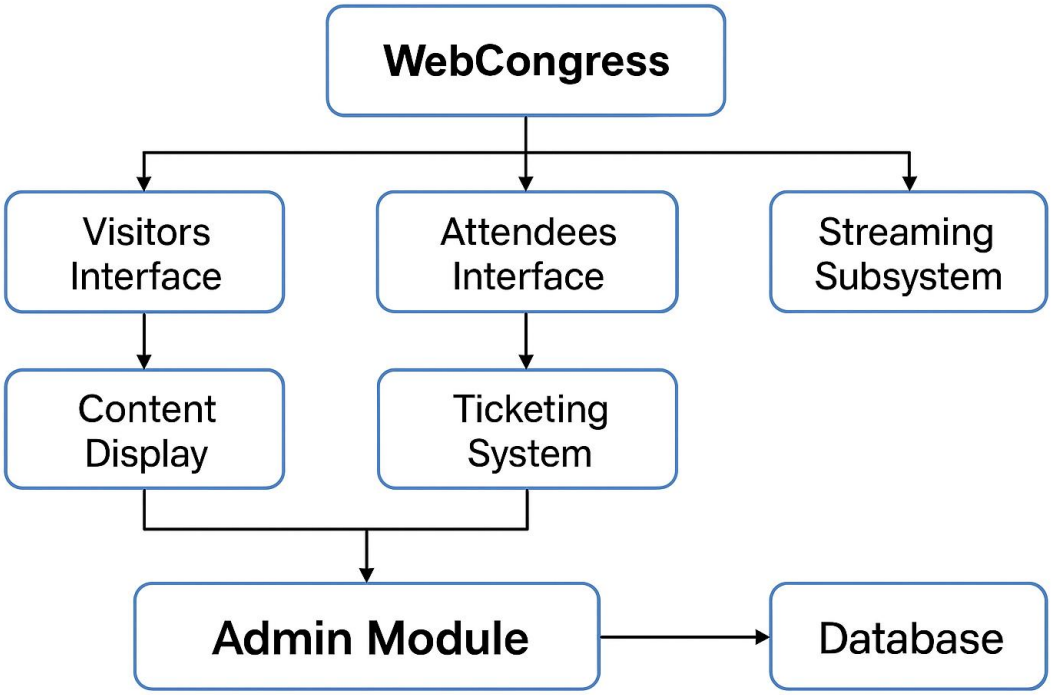

Figure 1 component diagram

4.3 Interface Design

The interface prioritizes simplicity and usability. All views are mobile-responsive and follow accessibility standards. The admin dashboard uses cards and tables for efficient data viewing.

- Navigation bar with sections: Events, Packages, Workshops/Conferences, Buy Pass
- Modal forms for registration and login
- Filters to search events by type or date
- Admin sidebar for event creation and analytics

4.4 Database Design

The system stores user data, event metadata, transactions, and attendance logs. Key tables include:

- users: stores personal and authentication data
- events: includes all event-related details (title, description, date, location)
- workshops: sessions within an event, with assigned speakers and capacity
- payments: tracks transactions with status, timestamp, and provider

Each entity has foreign key relationships to maintain data integrity.

4.5 Design Constraints

- Must support multilingual content if extended
- Must integrate securely with third-party payment APIs
- Should allow future extension for features like certificates, surveys, or session recordings

This design ensures flexibility, maintainability, and scalability as WebCongress evolves.

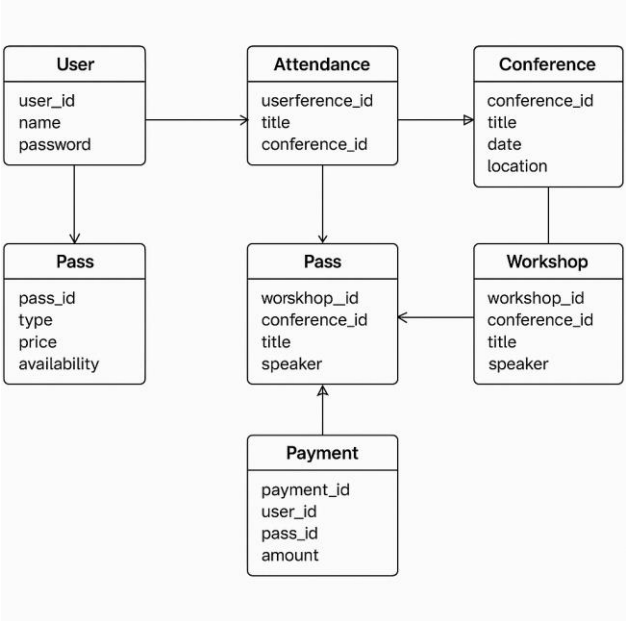

Figure 2 Entity relationship diagram

5. Test

This section details the testing strategy used in the development of the WebCongress system, including test types, tools, scenarios, and results.

5.1 Testing Strategy

The testing process followed a multi-layered approach, including:

- **Unit testing**: Validates individual functions and components.
- **Integration testing**: Ensures correct interaction between modules (e.g., user login with payment validation).
- **System testing**: Confirms the complete system behaves as expected.
- **Regression testing**: Re-runs previous test cases after changes (e.g., post-CR001 and CR002).
- **User acceptance testing (UAT)**: Performed by a test group simulating real usage scenarios.

5.2 Tools Used

- **Postman**: For API testing and response validation
- **PyTest**: For backend unit tests (Flask routes and services)
- **Jest (optional)**: For frontend logic testing
- **Manual testing**: For UI interaction and acceptance checks

5.3 Test Scenarios

Several key scenarios were tested:

ID	Scenario Description	Expected Result	Status
TC-001	User registration and login	New user created and authenticated	Passed
TC-002	Purchase of free and paid passes	Confirmation email and payment saved	Passed

TC-003	Chat functionality during live event	Real-time message delivery	Passed
TC-004	Admin creating and editing congress events	Data saved and reflected in frontend	Passed
TC-005	Integration with Stripe/PayPal APIs	Payment processed and verified	Passed
TC-006	SQL injection on login field	Input sanitized and rejected	Passed
TC-007	UI display on mobile screens	Responsive layout and readability	Passed

5.4 Test Coverage

Test coverage included all major modules:

- **Auth module**: 95% backend coverage
- **Event and registration module**: 90% coverage including UI paths
- **Payment integration**: 100% of API routes tested
- **Admin dashboard**: UI tested manually and validated by QA

Universidad de Guadalajara

Centro Universitario de los Valles

Web system for congress management

CRC-SIPaF-V2.2

Master's Degree in Software Engineering

Professor: Dr. Omar Ali Zatarain Durán

Author: Gutiérrez Constantino Roberto Carlos

Change control

Change Request ID	Requestor	Date	Status
CR-001	Dr. Omar Zatarain	03/14/2025	Approved
CR-002	Dr. Omar Zatarain	03/14/2025	Approved
CR-003	Dr. Omar Zatarain	03/14/2025	Approved

Revision

Change Request ID	Owner	Date	Status
1.0	Roberto Gutierrez	03/29/2025	100%

Change request 01

Objective of change: The goal of implementing a live streaming system for WebCongress conferences is to expand the event's reach, improve accessibility, and offer an interactive experience for attendees, enabling real-time participation without the need to be physically present at the conference.

State 1: Design Phase

If the implementation of the streaming system for live conferences is requested during the software design phase, the impact will be significant, but will be manageable.

System Component Impact

System Component	Impact
Backend Architecture	Streaming servers, video compression, and transmission protocols must be included
Database	Conference recordings and streaming event metadata must be managed
Frontend (User Interface):	Video players, live chat, and user controls must be added.
Administration Module:	Organizers must be able to manage live streaming and configure streaming events.

Impact on Cost: Total cost increase an increase of $2,500 to $3,500 USD is estimated due to the need for streaming servers and infrastructure adjustments.

Impact over Time: Estimated additional time: +1.5 to 2 months due to integration of new technologies and transmission testing.

Acceptance of change: Depending on the available budget and strategic priority, it can be accepted if the Decision Council approves the increase in costs and time.

State 2: Streaming System Implementation in the Mid-Development Phase (50% Completed)

If the change is requested when the software is already 50% developed, the impact will be significantly greater than if it had been considered in the design phase. The main reason is that architectures, databases, and workflows have already been defined, which can mean that streaming integration requires complex structural modifications.

System Component Impact

System Component	Impact
Backend Architecture:	Reconfiguration is required to support live streaming, which may affect already implemented modules.
Database:	Must be adjusted to store conference recordings, which may affect the current data structure
Frontend:	Pre-developed screens must be modified to include live video players, chat, and user controls.
Administration Module	Modification is required to allow administrators to manage streaming events.

Impact on Cost: Total cost increase an increase of $3,500 to $5,000 USD is estimated, as the change involves reconfiguring and rewriting parts of the system already implemented.

Impact over Time: Estimated additional time: +2 to 2.5 months due to code restructuring and compatibility testing.

Acceptance of change: NOT ACCEPTED, unless additional financing is obtained to absorb the additional cost.

State 3: Streaming System Implementation in the Final Phase (90% of Development Completed)

If the change is requested when the software is 90% developed, the impact will be critical because the architecture, databases and functionalities have already been implemented and tested.

System Component Impact

System Component	Impact
Backend Architecture:	Already tested code must be restructured, increasing the risk of failures in other parts of the system.
Database:	It must be modified to store recordings and handle real-time streaming, affecting the integrity of existing data.
Frontend:	Completed screens must be modified to add streaming features and additional controls.
Administration Module	Modifications to the administration panels are required, which could affect the stability of the platform in production.

Impact on Cost: Total cost increase: $5,000 - $7,000 USD due to the need to rework already developed and tested modules.

Impact over Time: Estimated additional time: +4 to 5 months, as completed modules must be modified and the entire system retested.

Acceptance of change: NOT ACCEPTED, It could only be considered for a future version of the system, since the impact on time, costs and structure is too high at this stage of the project.

STRENGHTS

- We have staff with the necessary experience and knowledge.

WEAKNESSES

- Increased demand for testing and optimization
- Requires a robust server infrastructure.

OPPORTUNITIES

- Possibility of integrating better tools

THREATS

- Limited budget

Change request 02

The goal of integrating more payment platforms into WebCongress is to expand the payment options available to users, improve financial accessibility, and increase sales by allowing more attendees to easily register and pay for their passes.

State 1: Integration of Payment Platforms in the Software Design Phase

If the integration of more payment platforms is requested during the software design phase, the impact will be low, since the payment system has not yet been implemented and adjustments can be made.

System Component Impact

System Component	Impact
Backend Architecture:	APIs should be designed to support multiple payment gateways.
Database:	More information about transactions, payment statuses, and methods used needs to be stored.
Frontend:	A responsive interface should be designed that allows users to choose between multiple payment options.
Administration Module	A section should be designed to monitor payments made through different providers.

Impact on Cost: Total cost increase $1,500 - $2,000 USD due to the need to integrate multiple third-party APIs and security testing.

Impact over Time: +1 per month as each payment gateway must be configured and extensive testing performed.

Acceptance of change: Change is accepted. In the design phase, it's easier to adjust the architecture to support multiple payment platforms without affecting work already done.

State 2: Integration of More Payment Platforms in the Mid-Stage of Development (50% Complete)

If the integration of more payment platforms is requested when the software is already 50% developed, the impact will be moderate to high.

System Component Impact

System Component	Impact
Backend Architecture:	Existing payment logic and controllers must be modified to support multiple gateways.
Database:	Additional information must be added to handle different states and payment methods.
Frontend:	Payment screens should be redesigned to include multiple options.
Administration Module	Options to view payments from different platforms should be included.

Impact on Cost: Total cost increase $2,500 - $4,000 USD, due to reconfiguration of already implemented modules and new compatibility testing.

Impact over Time: +2 months, as existing modules must be modified, tested, and compatibility with the current system must be ensured.

Acceptance of change: It is not accepted at this stage unless there is strategic justification and additional funding.

State 3: Integration of More Payment Platforms in the Final Phase (90% of Development Completed)

If the integration of more payment platforms is requested when the software is 90% developed, the impact will be critical.

System Component Impact

System Component	Impact
Backend Architecture:	Already tested and optimized code must be modified, which can introduce errors.
Database:	New payment method records must be added, affecting already structured financial reports.
Frontend:	Forms and payment flows must be redesigned, which could disrupt the already validated user experience.

Administration Module	Transaction management should be updated to allow monitoring of payments from different platforms.

Impact on Cost: Total cost increase $4,000 - $6,000 USD, as it requires reworking already tested and certified modules, in addition to performing additional testing.

Impact over Time: +3 to 4 months, as the entire payment integration must be reconfigured and tested without compromising system stability.

Acceptance of change: NOT ACCEPTED, It could only be considered for a future system update, after launch.

STRENGHTS

- We have staff with the necessary experience and knowledge.

WEAKNESSES

- Additional costs for payment gateway commissions.
- Greater complexity in transaction management and financial reconciliation

OPPORTUNITIES

- Possibility of integrating better tools

THREATS

- Limited budget
- Requires agreements with banking institutions

Change request 03

The goal of implementing a networking feature in WebCongress is to improve interaction between attendees, allowing them to establish professional connections and expand their network of contacts within the event.

State 1: Implementing a Networking Function in the Software Design Phase

If the implementation of the networking function is requested during the software design phase, the impact will be low.

System Component Impact

System Component	Impact
Backend Architecture:	A module must be designed to manage contacts, messaging and connection between users.
Database:	Storage of user profiles, connection requests and messages should be planned.
Frontend:	New sections should be designed for profiles, chat, networking recommendations, and attendee search.
Administration Module	Tools should be added to manage reports of misuse or moderate interactions.

Impact on Cost: Total cost increase $2,000 - $3,500 USD, due to the need to design and implement new data structures and interactive functionality.

Impact over Time: +1.5 to 2 months, as new interfaces, databases, and user connection logic must be designed.

Acceptance of change: Change is accepted. In the design phase, requirements can be defined from the outset, avoiding future rework. It's a strategic change that improves the user experience and increases the value of the system.

State 2: Implementing a Networking Function in the Mid-Phase of Development (50% Complete)

If the networking function is requested when the software is already 50% developed, the impact will be significant, since several modules of the system have already been designed and programmed

System Component Impact

System Component	Impact
Backend Architecture:	The system structure must be modified to allow connections between users and messaging.
Database:	New tables need to be added to handle contacts, connection requests, and messages.
Frontend:	New interfaces should be added to display profiles, contact lists, and chat options.
Administration Module	A tool is needed for administrators to moderate interactions and manage abuse reports.

Impact on Cost: Total cost increase: $3,500 - $5,000 USD, due to the need to modify already developed code and test integration with other modules.

Impact over Time: +2 to 2.5 months, as existing modules need to be modified and compatibility testing performed.

Acceptance of change: It is not accepted at this stage unless there is strategic justification and additional funding.

State 3: Implementation of a Networking Function in the Final Phase (90% of Development Completed)

If the networking function is requested when the software is already 90% developed, the impact will be critical, since modules that have already been completed, tested and optimized will have to be modified.

System Component Impact

System Component	Impact
Backend Architecture:	Already tested and optimized code must be modified to integrate networking logic.
Database:	New tables and relationships are required to store connections between users, messages, and networking preferences.
Frontend:	Completed interfaces must be modified and added to include profiles, contact lists, and chats.

Administration Module	A tool is needed to moderate interactions and manage abuse reports, which requires modifications to the admin panel.

Impact on Cost: Total cost increase $5,000–$7,500 USD, as it requires reworking completed modules, performing new tests, and ensuring compatibility with the existing system.

Impact over Time: +3 to 4 months, due to modification of tested code, integration with existing modules and stability testing.

Acceptance of change: NOT ACCEPTED. We recommend postponing this feature for a future system update.

STRENGHTS

- We have staff with the necessary experience and knowledge.

WEAKNESSES

- Additional storage may be required for user data

OPPORTUNITIES

- Possibility of integrating better tools

THREATS

- Limited budget
- Add secure privacy and security methods

Universidad de Guadalajara
Centro Universitario de los Valles

Web system for congress management

SAR-SIPaF-V0.1

Master's Degree in Software Engineering

Professor: Dr. Omar Ali Zatarain Durán
Author: Gutiérrez Constantino Roberto Carlos

Status Accounting Report

This section summarizes the implementation status of the approved change requests (CRs) within the WebCongress project, from planning to execution.

Change Request: CR-001 – Networking Module

- **Date of Approval:** 07/03/2025
- **Description:** Implementation of a real-time chat module for attendees during congresses.
- **Configuration Items Affected:** chatService.js, chatController.py, test cases, user manual
- **Involved Personnel:** Mariana López (Frontend), Raúl Soto (CM), Sofía Rivera (QA)
- **Estimated Budget:** $800
- **Real Budget:** $950
- **Estimated Time:** 10 days
- **Real Time:** 13 days
- **Human Resource Performance:** High collaboration; delays caused by concurrency issues in WebSocket implementation, resolved with help from backend support.
- **Issues:** Initial latency in chat messages and UI inconsistencies on mobile devices.
- **Actions Taken:** Redesigned message queue handling and applied responsive fixes.
- **Outcome: Solved**

Change Request: CR-002 – Payment Integration

- **Date of Approval:** 25/03/2025
- **Description:** Integration of payment gateways (Stripe and PayPal) for secure attendee pass purchases.
- **Configuration Items Affected:** stripeIntegration.py, paymentRouter.js, user manual, test suite
- **Involved Personnel:** David Hernández (Backend), Mariana López (Frontend), Sofía Rivera (QA)
- **Estimated Budget:** $950
- **Real Budget:** $1100
- **Estimated Time:** 12 days
- **Real Time:** 15 days
- **Human Resource Performance:** Efficient integration by backend team; slight delay due to misconfigured API keys in production.
- **Issues:** Payment confirmation failures during deployment.
- **Actions Taken:** Updated server configuration and added transaction logging.
- **Outcome: Solved**

Auditing Section

This section presents the audit results for the WebCongress system, including checklist validation and process compliance.

2.1 Audit Preparation and Objective

The objective of the audit was to confirm that all approved changes (CR001 and CR002) were implemented as planned, all deliverables were properly versioned, and no unauthorized changes were made. The audit covered functional and physical aspects.

Criterion	Result	Comments
Baseline matches implementation	Passed	Baseline v1.2 confirmed with matching CIs
Functional requirements met (FCA)	Passed	CR001 and CR002 passed all system tests
Physical configuration (PCA) verified	Passed	Deliverables matched documentation and repository contents
Test results documented and valid	Passed	All critical and non-critical tests executed and archived
No unauthorized changes detected	Passed	All updates were documented through CRs
Change requests fully traced and closed	Passed	CR tracking and approval process complete and recorded

2.3 Auditors

- Darcye Guerrero – Configuration Manager
- Ivan Faez – QA Lead
- Carlos Gutierrez – Project Manager

2.4 Audit Outcome

The WebCongress project passed both the Functional Configuration Audit (FCA) and the Physical Configuration Audit (PCA). All reviewed items were verified as compliant. Minor interface issues identified during the first review were resolved prior to delivery.

No discrepancies or unauthorized changes were found. The configuration management plan was followed as expected, confirming that the project met the quality and traceability standards established at the beginning of the development cycle.

Auditing Activities

This section describes the auditing tasks and timeline that were followed to verify the correct application of SCM practices during the WebCongress project.

4.1 Audit Planning

- Defined the scope of the audit (CR001 and CR002).
- Identified auditors and assigned responsibilities.
- Prepared audit checklists based on SCM plan.

4.2 Execution of Audits

- Conducted Functional Configuration Audit (FCA) to ensure requirements were met.

- Conducted Physical Configuration Audit (PCA) to verify actual implementation matched documentation.
- Reviewed source code, documents, test reports, and baseline traceability.

4.3 Evidence Collection

- Collected test execution reports, screenshots, commit logs, and approval forms.
- Compared repository files with approved baselines (v1.2).

4.4 Findings and Resolutions

- Found minor interface inconsistencies, resolved before final delivery.
- Confirmed all changes were authorized and documented.

4.5 Final Review

- Audit report generated and signed by auditors.
- Results shared with project stakeholders.
- Audit files archived.

These activities ensured that the system was delivered in a compliant, traceable, and verifiable state, fully aligned with the SCM process defined at the beginning of the WebCongress project.

References

- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.
- Sommerville, I. (2011). *Software Engineering* (9th ed.). Pearson.
- IEEE Standard for Software Configuration Management Plans. IEEE Std 828-2012.
- ISO/IEC/IEEE 12207:2017 – Systems and software engineering – Software life cycle processes.
- Git documentation: https://git-scm.com/docs
- Stripe API documentation: https://stripe.com/docs/api
- PayPal Developer documentation: https://developer.paypal.com/
- Flask framework documentation: https://flask.palletsprojects.com/