EDUCACIÓN
IPSUMTEC
ISSN: 2594 - 2905

WEB-BASED EVENT MANAGEMENT SYSTEM WITH A DIGITAL CALENDAR USING PYTHON OSS TECHNOLOGIES AND REPORT VALIDATION

WEB-BASED EVENT MANAGEMENT SYSTEM WITH A DIGITAL CALENDAR USING PYTHON OSS TECHNOLOGIES AND REPORT VALIDATION

López Aburto María Teresa[1] , García López Serbando[2] , Gonzalez Guarneros Filiberto[3] , Martínez Ramírez Violeta[4] , Pérez Irigoyen Miguel Ángel[5]

[1] National Technological Institute of Mexico/Puebla Technological Institute,mariateresa.lopez@puebla.tecnm.mx , https://orcid.org/0009-0001-9300-067
[2] National Technological Institute of Mexico/Technological Institute of Puebla,serbando.garcia@puebla.tecnm.mx , https://orcid.org/0009-0001-9484-9587
[3] National Technological Institute of Mexico/Puebla Technological Institute,filiberto.gonzalez@puebla.tecnm.mx , https://orcid.org/0009-0002-2078-7376
[4] National Technological Institute of Mexico/Puebla Technological Institute.violeta.martinez@puebla.tecnm.mx , https://orcid.org/0000-0003-1518-786X
[5] National Technological Institute of Mexico/Puebla Technological Institute,i17220636.19@puebla.tecnm.mx , https://orcid.org/0009-0003-4246-7290

https://doi.org/10.61117/ipsumtec.v8i2.426

Abstract – This paper presents the technical development of the "Web-based System for Event Management and Scheduling for the Extracurricular Activities Department at the Technological Institute of Puebla." The purpose of the system is to allow the departments of the Institute to request spaces for extracurricular events, control availability, manage events, and generate monthly and quarterly reports that include graphic evidence. A client-server architecture based on modern web technologies such as Python with Flask, MariaDB, and the FullCallendar calendar plugin was used, in addition to tools such as wkhtmltopdf for report generation.

Keywords-- Flask, Python, FullCallendar wkhtmltopdf, MariaDB.

Abstract -- This paper presents the technical development of the "Web-based System for Event Management and Scheduling for the Extracurricular Activities Department of the Instituto Tecnológico de Puebla." The system's purpose is to allow departments at the Instituto Tecnológico de Puebla to request spaces for extracurricular events, monitor availability, manage events, and generate monthly and quarterly reports that include graphical evidence.
A client-server architecture based on modern web technologies such as Python with Flask, MariaDB, and the FullCallendar calendar plugin was used, as well as tools such as wkhtmltopdf for report generation.

Key words **–** Flask, Python, FullCallendar wkhtmltopdf, MariaDB.

INTRODUCTION

The system was developed in the context of the Technological Institute of Puebla, specifically within the Planning and Outreach Division, in collaboration with the Extracurricular Activities Department. This area is responsible for coordinating, authorizing, and scheduling extracurricular events involving the technological community, both internally and externally.
During its development, the need to automate and optimize the event management process was identified, as it was previously carried out through a channel on the Microsoft Teams platform and physical formats. This led to redundancy in information, delays in event approval, and poor visibility of the global calendar.
The work environment consisted of a mixed team of administrative staff, teachers in charge of activities, and social service or professional residency residents. Constant interaction with real users of the system (administrators and requesting users) provided continuous feedback to improve the usability of the developed software.
In addition, we worked in a semi-schooled environment, which allowed us to combine face-to-face and virtual sessions, as well as local and institutional server testing. This flexibility was key to meeting the established deadlines and facilitating system testing prior to its final implementation.
Problem Statement
At the Puebla Institute of Technology, the Extracurricular Activities Department is responsible for coordinating various academic, cultural, sporting, and recreational events that enrich the students' comprehensive education.

However, these events are currently managed manually or using non-standardized methods, which creates several problems, such as:

Receipt: October 17, 2025 | Acceptance: November 22, 2025 | Publication: December 19, 2025

- Difficulty in efficiently tracking event requests.
- Potential conflicts in the allocation of dates and spaces.
- Lack of clear visibility on already scheduled events.
- Delays in responding to requests.

This situation can lead to institutional disorganization, duplication in the use of spaces, and wasted time for both users and administrative staff. Therefore, a technological solution is required to manage event requests and scheduling more efficiently, as well as to provide a clear and accessible view of the institutional extracurricular activities calendar.

General Objective:

Design and implement a web-based system to manage requests and approvals for institutional events with PDF report generation.

Specific Objectives:

1. Implementation of a dynamic form for capturing event data.
2. Create an administrative panel for managing user requests.
3. Prevent event overlap to avoid conflicts of location and dates.
4. Generate monthly and quarterly reports dynamically with events divided by week and in chronological order.
5. Upload images to approved events to add as visual evidence in reports.

Scope

The main purpose of the system developed is to facilitate event management within the Puebla Institute of Technology, specifically in the Extracurricular Activities Department. The main features of the system are detailed below:

It allows registered users (teachers and administrative staff) to request events through an integrated form, which includes fields such as event name, start and end dates, location, description, event type (internal or external), and person responsible.

Administrators can access a specialized view from which they can approve or reject events, as well as view a summary of scheduled events in an interactive calendar.

The system has a control mechanism to prevent duplicate events at the same place and time, automatically blocking spaces that have already been assigned and approved.

Once an event is approved, the user can upload multiple images associated with it, which will be used to generate monthly and quarterly reports in PDF format.

A role and authentication system has been implemented, where users can only access the functionality corresponding to their profile (user or administrator).

The system automatically generates PDF reports, organizing events by week and grouping relevant information, including images, to facilitate their presentation in institutional settings.

The system has been developed with open source technologies: Python, Flask, SQLite (and optional connection to MariaDB on the server), JavaScript (with FullCalendar), HTML/CSS, and the Jinja2 template engine.

It can be deployed locally for testing and is also ready to be uploaded to a production server such as Render or implemented in XAMPP with a connection to MariaDB.

Python is a highly sought-after interpreted language for web application development due to its efficiency, ease of learning, and multi-platform capabilities [1].

Web development in Python is simplified thanks to lightweight frameworks such as Flask. The Flask API allows you to create web applications quickly with little repetitive code. [2]

FullCalendar displays data in calendar view. It is an excellent option for displaying a standard SharePoint calendar, extracting data from multiple calendar lists, data from non-calendar lists, or even data external to SharePoint. [3].

SQLite is a relational database management system integrated into multiple servers and web applications, making it easy to configure and connect to Maria DB [4].

Jinja templates in Python offer flexibility and ease of use, allowing it to stand out as a powerful tool for dynamic content creation in web applications [5]. Templates are usually composed only of variables and/or expressions, which are replaced with values when rendered and control the logic in them. Its syntax is inspired by Django and Python. [6]

Wkhtmltopdf defines itself as an open-source command-line tool for converting HTML to PDF documents [7].

State of the Art

There are studies available on the internet about the combination that can be made with server-side web applications, which combine frameworks for Python, among the most popular are Pyramid, Flask, and Bottle. It is claimed that Python and Django optimize the time spent on backend application production, as Django's structure provides more reliable security properties. This significantly reduces the cost of investment in technological development for public institutions.

In Colombia, the published work "Development of a Python-based web interface for real-time acoustic speech analysis" demonstrates the ease of creating a web application based on Python tools that successfully analyzes audio in real time and offline. The interface, designed with wireframes, made it possible to identify the user interaction flow and

Receipt: October 17, 2025 | Acceptance: November 22, 2025 | Publication: December 19, 2025

structured backend processing thanks to Flask and Parselmouth for audio analysis [8].

The project "Design and implementation of an employee news report management system based on a web application" developed in Colombia also chose Python as the main language for creating the system that manages the process of reporting results of licenses, disabilities, retirements, and employee transfers within a company. Excellent automation was achieved for better decision-making. [9]

In Ecuador, the academic control system developed in a web environment was based on Python with Bootstrap4 and the Flask framework. HTML templates were used to create a user-friendly and dynamic interface that facilitated web interaction. The results were published under the title "Python and Cloud Computing-Based Academic Control System at the Secondary Level in the City of Santo Domingo." [10]

The work entitled "Web system based on association rules to support the sales process at Ayala Motors, 2022" shows an implementation of the web *recommendation* system, using tools such as Google *Colab*, Python, Visual Studio Code, PHP, Laravel, and PostgreSQL, which made it possible to develop a technological tool that will effectively support employees in the sales process within the company [11].

DEVELOPMENT

The system was developed locally in the following environment:

- Main Language: Python
- Framework: Flask
- Database: MariaDB (XAMPP)
- Frontend: HTML5, CSS3 (custom) + FullCallendar
- PDF Reports: wkhtmltopdf
- IDE: Visual Studio Code
- Final server: Windows Server

Development methodology

The agile approach allowed for rapid iteration on the development of key features such as the events calendar, user registration and authentication, request management, administrator approval/rejection, and monthly and quarterly report generation.

Each iteration included:

- Review of requirements with the advisor.
- Implementation of a specific functionality.
- Testing and immediate feedback.
- Adjustments based on feedback received.

Technologies used

- Back-end: Flask Framework (Python), with a modular structure based on routes, models, and templates. Flask-Login was used for authentication and session control.
- Front-end: HTML5, CSS3, and JavaScript, with support from the FullCalendar.js library for the dynamic and interactive calendar.
- Database engine: MariaDB for production environments.
- Dynamic templates: Jinja2 was used to build dynamic and customized views.
- PDF report generation: PDFKit was used together with wkhtmltopdf to automatically convert HTML templates into PDF files.
- Development environment: Visual Studio Code on Windows 10, with Python virtual environment (venv) and local execution with Flask. The system was prepared for deployment on a Windows Server with XAMPP.

Event creation and management code solicitar_evento: This function allows new events to be registered from the user or admin form. The event remains pending approval until it is reviewed by an admin for future approval or rejection for correction:

Image **1**. *Request events*.

Source: *Own work* (2025).

Part 1 of the solicitar_eventos block, where the data entered in the form is sent and the dates are checked for consistency. See image 1.

Part 2 of solicitar_eventos, where, before saving the event in the database, it is checked whether there is already an approved event with the same location and date range. See image 2.

Image 2. *Request events after validation*.

Source: *Own creation* (2025).

Part 3 of request_event shows the user that their event has been sent to the administrator correctly for approval or rejection. See image 3.

Image 3. *Event request is approved or rejected*.

Source: *Own creation* (2025).

Event management code

"Approve" block: when the administrator approves an event, they send it to the database to be displayed on the calendar and allow images to be uploaded for PD reports. See image 4.

Image 4. *Validate and display calendar*.

```python
@eventos.route('/aprobar/<int:evento_id>')
@login_required
def aprobar_evento(evento_id):
    evento = Evento.query.get_or_404(evento_id)
    evento.aprobado = True
    evento.motivo_rechazo = None
    db.session.commit()
    flash(f'Evento "{evento.nombre}" aprobado.', "success")
    return redirect(url_for('eventos.admin_index'))
```

Source: *Own creation* (2025).

"Reject" block: when rejecting an event, the administrator must enter a reason, which will be displayed to the user so that corrections can be made if necessary and the event resubmitted. See image 5.

Image 5. *Reject events*.

```python
@eventos.route('/rechazar/<int:evento_id>', methods=['POST'])
@login_required
def rechazar_evento(evento_id):
    motivo = request.form.get('motivo_rechazo')
    if not motivo:
        flash("Debes ingresar un motivo para el rechazo.", "danger")
        return redirect(url_for('eventos.admin_index'))

    evento = Evento.query.get_or_404(evento_id)
    evento.aprobado = False
    evento.motivo_rechazo = motivo
    db.session.commit()
    flash(f'Evento "{evento.nombre}" rechazado.', 'warning')
    return redirect(url_for('eventos.admin_index'))
```

Source: *Own creation* (2025).

Report generation code

The following code block shows how the system generates monthly and quarterly reports: Part 1 of the reporte_mes block, which checks which month is being viewed in the calendar when generating the prf, as well as checking if there are events in the month, generating the title, and checking if there are images in the folder to be considered in the generation cycle. Part 2 of the reporte_mes block, where the information is processed to generate the PDF. See image 6.

Image 6. *Generate reports*.

Source: *Own work* (2025).

Part 1 of the reporte_trimestre_rango block, where the initial date of the report is calculated based on the month being viewed on the calendar, as well as obtaining events. See image 7.

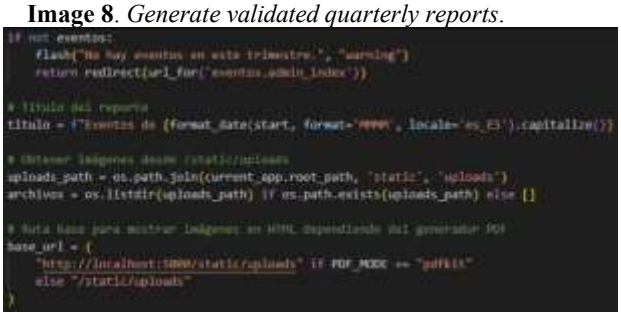
Image 7. *Generate quarterly reports*.

```python
@eventos.route('/reporte_trimestre_rango/<int:anio_inicio>/<int:mes_inicio>')
@login_required
def reporte_trimestre_rango(anio_inicio, mes_inicio):
    if mes_inicio < 1 or mes_inicio > 12:
        flash("Mes de inicio inválido para el trimestre.", "danger")
        return redirect(url_for('eventos.admin_index'))

    # Fecha de inicio
    start = datetime(anio_inicio, mes_inicio, 1)

    # Calcular fecha final
    mes_fin = mes_inicio + 2
    anio_fin = anio_inicio
    if mes_fin > 12:
        mes_fin -= 12
        anio_fin += 1
    end = datetime(anio_fin, mes_fin, monthrange(anio_fin, mes_fin)[1], 23, 59)

    # Obtener eventos aprobados en el rango
    eventos = Evento.query.filter(
        Evento.aprobado == True,
        Evento.fecha_inicio >= start,
        Evento.fecha_inicio <= end
    ).order_by(Evento.fecha_inicio).all()
```

Source: *Own creation* (2025).

Part 2 of the reporte_trimestre_rango block, where it is checked whether there are any events in the month, the report title is generated, and the images are verified to be considered in the generation. See image 8.

Image 8. *Generate validated quarterly reports*.



Source: *Own work* (2025).

Receipt: October 17, 2025 | Acceptance: November 22, 2025 | Publication: December 19, 2025

Part 3 of the reporte_trimestre_rango block, where all the information obtained for the report is assigned, and the existence of the wkhtmltopdf generator is verified. See image 9.

Image 9. *Generate reports by range*.

```
html = render_template(
    "reporte_pdf.html",
    eventos=eventos,
    titulo=titulo,
    archivos=archivos,
    timedelta=timedelta,
    format_date=format_date,
    os_path=os.path.abspath(os.curdir),
    es_trimestral=True,
    imagen_base_url=base_url
)

# Generar el PDF
if PDF_MODE == "weasy":
    pdf = HTML(string=html, base_url=current_app.root_path).write_pdf()
elif PDF_MODE == "pdfkit":
    options = {
        'enable-local-file-access': None
    }
    pdf = pdfkit.from_string(html, False, configuration=config, options=options)
else:
    flash("No se pudo generar el PDF. Verifica la configuración.", "danger")
    return redirect(url_for('eventos.admin_index'))
```

Source: *Own creation* (2025).

Part 4 of the reporte_trimestre_rango block, where the report is generated for viewing and downloading.

Image 10. *Downloading the report in PDF* format.

```
# Devolver el PDF como respuesta
response = make_response(pdf)
response.headers['Content-Type'] = 'application/pdf'
response.headers['Content-Disposition'] = f'inline;
return response
```

Source: *Own creation* (2025).

Image upload code

Below is the code where the images to be uploaded are processed. Different formats are accepted and they are assigned a name according to the event:

Part 1 of the subir_imagenes block, where images are processed to determine whether they are valid, and names are assigned according to the name of the event.

Image 11. *Image upload process*.

```
@eventos.route('/subir_imagenes/<int:evento_id>', methods=['POST'])
@login_required
def subir_imagenes(evento_id):
    evento = Evento.query.get_or_404(evento_id)

    if evento.usuario_id != current_user.id or not evento.aprobado:
        flash("No puedes subir imágenes para este evento.", "danger")
        return redirect(url_for('eventos.user_index'))

    archivos = request.files.getlist('imagenes')
    if not archivos or archivos[0].filename == '':
        flash("No se seleccionó ninguna imagen.", "warning")
        return redirect(url_for('eventos.user_index'))

    extensiones_permitidas = {'png', 'jpg', 'jpeg', 'gif', 'webp'}
    rutas_guardadas = []

    uploads_dir = os.path.join(current_app.root_path, 'static', 'uploads')
    os.makedirs(uploads_dir, exist_ok=True)

    nombre_base = limpiar_nombre(evento.nombre)

    # Buscar cuántas imágenes ya existen con este nombre base
    archivos_existentes = [
        f for f in os.listdir(uploads_dir)
        if f.startswith(f"evento_{nombre_base}_")
    ]
```

Source: *Own work* (2025).

Part 2 of the subir_imagenes block, in this part the image(s) are listed as they are checked to see if there is already one previously and then uploaded to the uploads folder. See image 11.

Image 11. *Image upload process*.

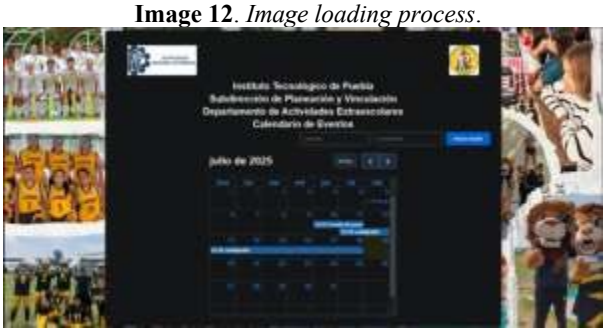
```
contador_inicial = len(archivos_existentes) + 1

for i, archivo in enumerate(archivos, start=contador_inicial):
    if archivo and '.' in archivo.filename:
        extension = archivo.filename.rsplit('.', 1)[-1].lower()
        if extension not in extensiones_permitidas:
            continue

        nombre_archivo = f"evento_{nombre_base}_{i}.{extension}"
        ruta_completa = os.path.join(uploads_dir, nombre_archivo)
        archivo.save(ruta_completa)

        rutas_guardadas.append(f"static/uploads/{nombre_archivo}")

if rutas_guardadas:
    evento.imagen = rutas_guardadas[-1]  # Última subida como principal
    db.session.commit()
    flash(f"Se subieron {len(rutas_guardadas)} imagen(es) correctamente.", "success")
else:
    flash("No se subió ninguna imagen válida.", "warning")

return redirect(url_for('eventos.user_index'))
```

Source: *Own creation* (2025).

Main screen:

This screen includes the login section for users and administrators, as well as the calendar where all events requested by users and approved by the administrator are displayed. Main screen of the system, displaying the institute's logos, the login section, and the calendar with events. See image 12.

Image 12. *Image loading process*.



Source: *Own work* (2025).

Main view of the administrator

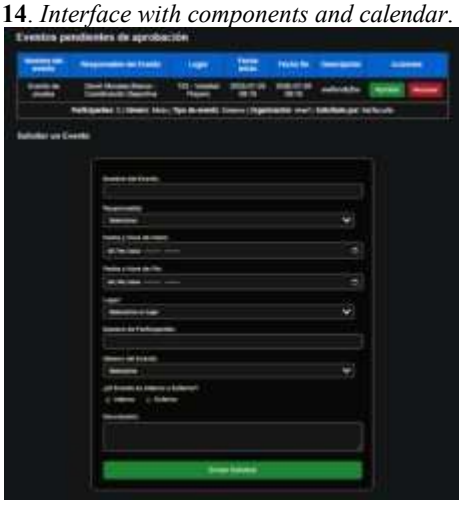
On this screen, the administrator can view the event calendar, approve requests, delete or reject events, and generate monthly and quarterly reports: Part 1 of the administrator screen includes the calendar, buttons for dynamically generating reports, and a button to delete all events in the month being viewed. See image 13.

Image 13. Interface with components and calendar.

Source: *Own work* (2025).

Part 2 of the administrator screen displays a section with pending events requested by users or created by the administrator. Next to this section are buttons to accept or reject events, and below is a form that allows the administrator to create their own events if necessary. See image 14.

Image 14. *Interface with components and calendar*.



Source: *Own work* (2025).

User view

Regular users can request events using a form and check the status of their events:

Part 1 of the user screen shows a logout button and the calendar with events.

Image 15. *Event calendar and exit*.

Source: *Own creation* (2025).

Part 2 of the user screen, section of events requested by the user, has edit buttons to correct errors in the information or a delete button to delete the event if necessary. See image 16.

Image 16. *Event calendar and logout*.

Source: *Own creation* (2025).

Part 3 of the user screen, a form for creating events, contains important information such as the name of the event, dates, location, participants, and whether the event was requested internally or externally, as well as a description section for clarifications. See image 17.

Image 17. *Event calendar and output*.

Source: Author's *own work* (2025).

DISCUSSION AND ANALYSIS OF RESULTS

Report generation

The system allows you to generate both monthly and quarterly reports in PDF format. These reports are temporary and are deleted when you return to the previous screen to prevent the system memory from becoming saturated. To preserve the content, the PDF must be downloaded:

Monthly report PDF, including the title with the month and each event in chronological order, divided by a line for visual clarity. See image 18.

Image 18. *Monthly report*.

Source: *Own work* (2025).

Display of calendar event data

Clicking on an event in the calendar opens a side window on the right side of the browser containing detailed information about the event or events for the selected day: Placing the mouse pointer over an event displays individual information about that event. See image 19.

Image 19. *Event details*.

Source: *Own work* (2025).

Clicking on any event will open a side window containing all the information about the event. This window can be closed by simply clicking outside it or by clicking on the "X" icon at the top right of the panel. See image 20.

Image 20. *Event details*.

Source: *Own creation* (2025).

Results

The result of the project is a functional, intuitive, and lightweight system that digitizes and automates the process of registering, controlling, and documenting institutional events, which was previously done manually or using basic forms.

Main achievements and benefits of the system

- Clear and accessible interface, allowing events to be viewed on a monthly basis and interactively by simply clicking on a date.
- Effective differentiation of roles, allowing the user and administrator experience to be customized from the moment of login.
- Automatic validation of requests, preventing duplicate locations or schedule conflicts.
- Smart blocking of dates and locations already occupied by approved events.
- Clear statuses for each request: approved, rejected (with visible reason), editable, or deletable in case of rejection.
- Upload and manage images by event to document activities and provide supporting documentation.
- Automatic PDF reports, organized by week, with integrated images and useful data for archiving or institutional presentations.
- Stable performance in a local environment, with good response speed and low difficulty level for new users.
- System scalability, ready to run on servers such as XAMPP or in the cloud, thanks to its modular architecture and use of widely compatible technologies (Table 1).

Table 1. *Comparative data*.

Criterion	Previous process (Teams channel + office of coordination)	Current process (Web system)
Application of application	Messages in a channel Microsoft Teams channel and physical delivery of formats.	Online form integrated into the system's website.
Approval time Approval time	Variable; depended on staff availability and manual review.	Immediate upon receipt; automatic validation of data and availability.
Schedule control	Manual, with risk of duplication of places and dates.	Automatic, with blocking of places and dates already occupied.
Event visibility	Limited; required consultation with the coordinator or the Teams channel.	Total; interactive calendar accessible for users and administrators.
Change management Change management	It required direct communication with the coordination and possib r forwarding of formats.	Online editing of events no t approved and d automatic resubmission for review.
Evidence logging	No There was a centralized method; it depended on mailing or courier.	Cargo of images directly to the event approved for reports.
Generation	Manual	Automatic in

Receipt: October 17, 2025 | Acceptance: November 22, 2025 | Publication: December 19, 2025

	text documents or presentations.	quarterly, with images and .
Access and security	Without control de roles; any person in the channel could see requests.	Access control by roles (user and administrator) with authentication.

Source: *Own work* (2025).

Functional testing

Throughout the development process, continuous functional tests were carried out, focusing on each individual module and on the overall integration of the system.

Tests performed

- Simulation of different types of users and authentication tests.
- Verification of form fields, valid dates, and detection of location conflicts.
- Review of the complete event approval and rejection flow.
- Dynamic interface behavior testing (e.g., showing/hiding buttons depending on event status).
- Report generation and detailed review of the content, format, and quality of inserted images.

These tests allowed minor errors to be detected at an early stage, which were corrected in each iteration cycle. At the end of development, the system demonstrated stable operation in all implemented functions, meeting the requirements established at the beginning of the project.

Discussion

With Amazon cloud technology (AWS), its expertise allows us to affirm that Python offers great ease for the implementation of multi-platform systems across different computer operating systems such as Windows, macOS, Linux, and Unix, in addition to strong data protection when sending information over the network thanks to complex functions available for web development. [1]

Having a digital calendar included in a web system allows users to easily access it from anywhere in the world and manage important events for the use of time and space without overlaps. Schedule coordination is essential for all concurrent users [13].

John Hill [14] describes "Calendars are essential for documenting and tracking important dates and events in our lives," emphasizing that project management is essential for both users and those involved in the administrative process to stay up to date on the status of each event.

By automatically obtaining digital reports, time is reduced by up to 30% and paper use is eliminated entirely.

In addition, information is centralized and ready for interpretation, promoting informed decision-making and, as a result, communication becomes more fluid and immediate institutionally. Data remains available and retrievable in real time, making it much easier to ensure the integrity of information throughout the process. [15]

CONCLUSIONS

The development of the event management system for the Extracurricular Activities Department of the Technological Institute of Puebla represented a comprehensive experience in the complete software development cycle. Throughout the project, good software engineering practices were applied, from problem analysis and requirements definition to implementation and functional validation with real users.

One of the main strengths of the system is its user-centered approach, allowing both administrative staff and general users to manage events clearly and securely. The use of Flask as the main framework facilitated a modular and efficient architecture, while technologies such as SQLite/MariaDB, HTML/CSS/JS, FullCalendar, and PDFKit made possible an intuitive interface and robust functionalities such as multiple image uploads, automatic report generation, and event status control.

Likewise, deployment in a controlled environment and compatibility with platforms such as XAMPP (MariaDB) and Render for production ensure the portability and adaptability of the system in future real-world scenarios.

During development, several challenges arose, such as managing static file paths in different environments, validating data from the front-end and back-end, and the logic for maintaining integrity in simultaneous event requests. All of these were successfully addressed by applying an incremental methodology based on agile philosophy, constantly validating progress with the project advisor.

Finally, the system not only meets the functional objectives set at the outset, but is also prepared for future scalability, such as the generation of quarterly reports, integration with institutional authentication systems, and visual improvements. The experience gained in this project strengthens the developer's capabilities at both the technical and organizational levels, laying the foundation for future work of greater complexity.

Future work

A future line of research could focus on the development of advanced interactive reporting systems that integrate dynamic data visualization, predictive analytics, and decision support tools. This approach would allow us to study how the use of graphics, smart filters, and automated infographics contributes to improving the accuracy of

Receipt: October 17, 2025 | Acceptance: November 22, 2025 | Publication: December 19, 2025

institutional projections and optimizes resource management [16].

Another line of research could focus on the use of Python as the main language for data analysis applied to web development. The research could address the integration of libraries such as Pandas, NumPy, and TensorFlow in server environments to build applications capable of processing and visualizing large volumes of information in real time [17].

BIBLIOGRAPHY

[1] Amazon Web Services. What is Python? [Internet]. Amazon; 2024 [cited Aug. 27, 2025]. Available from: https://aws.amazon.com/es/what-is/python/

[2] Ramotion. Python for web development [Internet]. 2025 [published July 22, 2025, cited August 27, 2025]. Available at: https://www.ramotion.com/blog/web-development-in-python/#:~:text=Simplicity%20and%20ease%20of use, with minimal code.

[3] Microsoft Learn. Migrate your jQuery and FullCalendar solution built with the Script Editor web part to SharePoint Framework [Internet]. 2022 [published June 29, 2022, cited Aug. 27, 2025]. Available at: https://learn.microsoft.com/es-es/sharepoint/dev/spfx/web-parts/guidance/migrate-jquery-fullcalendar-script-to-spfx

[4] Hassan A. What is SQLite? [Internet]. BUILTIN; 2025 [published 2025 Apr 24, cited 2025 Aug 27]. Available at: https://builtin.com/data-science/sqlite#:~:text=Benefits%20of%20SQLite,separate%20code%20in%20another%20language.

[5] Atareao. The power of Jinja and Python [Internet]. 2024 [published 9 Apr 2024, cited 27 Aug 2025]. Available at: https://atareao.es/podcast/el-poder-de-jinja-y-python/#:~:text=About%20the%20advantages%20of%20using,specific%20needs%20of%20your%20projects.

[6] Jinja Project. Template designer documentation [Internet]. 2007 [cited Aug. 27, 2025]. Available at: https://jinja.palletsprojects.com/en/stable/templates/

[7] WKHTMLTOPDF. What is it? [Internet]. [n.d.] [cited Aug. 27, 2025]. Available at: https://wkhtmltopdf.org/

[8] Molina Giraldo MA, Oñate González AF. Development of a Python-based web interface for real-time acoustic speech analysis [Internet]. Bogotá: Pontificia Universidad Javeriana; 2024 [cited 27 Aug 2025]. Available at: https://repository.javeriana.edu.co/items/227d28c0- c814-4348-b895-8402d8e30972

[9] López Bran DA, Montoya Sáenz A. Design and implementation of an employee incident report management system based on a web application [Internet]. Rev Univ Catol Oriente. 2020;31(45):28-44 [cited 28 Aug 2025]. Available from: https://revistas.uco.edu.co/index.php/uco/article/view/28 1

[10] Tejada Campos DA. Academic control system based on Python and cloud computing, at the secondary level in the city of Santo Domingo [Internet]. Santo Domingo (Ecuador): Universidad Regional Autónoma de los Andes; 2020 [cited 28 Aug 2025]. Available at: https://dspace.uniandes.edu.ec/handle/123456789/11358

[11] Onofre Ávila BA. Analysis of web applications using Python for backend development in public institutions [Internet]. Babahoyo (Ecuador): Technical University of Babahoyo; 2021 [cited Aug. 28, 2025]. Available at: https://dspace.utb.edu.ec/handle/49000/9534

[12] de Iman UA, Lila N. Web system based on association rules to support the sales process of the Ayala Motors company [Internet]. Chiclayo (Peru): Santo Toribio de Mogrovejo Catholic University; 2022 [cited Aug. 28, 2025]. Available at: http://hdl.handle.net/20.500.12423/7815

[13] Doodle. Advantages of a digital agenda: saving time at work [Internet]. 2025 [published Mar 24, 2025, cited 7 Sep 2025]. Available at: https://doodle.com/es/benefits-of-a-digital-calendar-saving-time-in-the-workplace/

[14] Hill J. The advantages of having a digital calendar over a paper calendar [Internet]. Calendar.com; 2023 [published 16 Mar 2023, cited 7 Sep 2025]. Available at: https://www.calendar.com/blog/the-advantages-of-having-a-digital-calendar-over-a-paper-calendar/#:~:text=Appointments%20can%20be%20moved,as%20if%20they%20had%20always%20been%20there.

[15] Kizeo Forms. Digital reports: uses and benefits for your company [Internet]. [n.d.] [cited 7 Sep 2025]. Available at: https://www.kizeo-forms.com/es/blog/informes-digitales-usos-y-beneficios-para-tu-empresa/#:~:text=Advantages%20of%20digital%20reports,time%20by%20up%20to%2030%.

[16] Duek C. Digital reports are a global trend worldwide [Internet]. Kaleida Digital; 2024 [published June 23, 2020, cited September 7, 2025]. Available at: https://kaleidadigital.com/es/los-reportes-digitales-son-una-tendencia-a-nivel-mundial/#:~:text=The%20benefits%20of%20reports,to%20companies%20and%20organizations.

[7] Brainvire Infotech Inc (2025). Exploring Python's leadership role in data analysis and its key advantages. [Internet] [published September 10, 2024, accessed November 9, 2025]. Available at: https://www.brainvire.com/blog/python-data-analytics-innovation/#:~:text=from%20other%20languages.-,Performance%20and%20scalability,big%20data%20analysis.

Table of Contribution Roles

Role	Author(s)
Conceptualization	Miguel Ángel Pérez Irigoyen

Data curation	Miguel Ángel Pérez Irigoyen
Methodology	Filiberto Gonzalez Guarneros
Project management	María Teresa López Aburto
Resources	Review and editing – Violeta Martínez Ramírez
Software	Miguel Ángel Pérez Irigoyen
Supervision	Serbando Garcia Lopez
Validation	Miguel Ángel Pérez Irigoyen
Visualization	María Teresa López Aburto
Writing	Original draft – Violeta Martínez Ramírez
Editing	Revision and editing – Violeta Martínez Ramírez

.