
UNISAFE SYSTEM ANALYSIS

Feasibility study and analysis



www.bittimeprofessionals.com

Table Of Contents

INTRODUCTION.....	3
DEFINITION OF EMERGENCY.....	3
PRESENT HANDLING OF EMERGENCIES.....	3
OUTGOING STUDENTS: FROM UNIPV TO ABROAD.....	3
INCOMING STUDENTS: FROM ABROAD TO UNIPV.....	4
PROJECT'S NEEDS AND GOAL.....	4
FINAL OUTPUTS.....	4
MAJOR STEPS OF THE FEASIBILITY STUDY.....	5
GENERAL ANALYSIS OF THE PROCESS.....	5
DETAILED ANALYSIS OF THE PROCESS.....	5
MULTICHANNEL COMMUNICATION.....	5
FEASIBILITY STUDY.....	6
TECHNOLOGY SELECTION.....	6
MACRO COMPONENT 1: MONITORING.....	8
MACRO COMPONENT 2: HUMAN SCR EVALUATION.....	9
MACRO COMPONENT 3: NOTIFICATION.....	10
Feasibility Study.....	11
Data Flow Analysis.....	11
Data Flow Monitoring Phase.....	11
Earthquakes Monitor.....	12
Twitter Monitors.....	12
Forward Geocoder Service.....	12
Localized Events Storer.....	13
PostgreSQL.....	13
POSTGRESQL INEXHAUSTIVE FEATURES LIST.....	14
Message Broker - NATS.....	15
Events Evaluator.....	16
Human SCR Evaluation Phase Data Flow.....	16
WebSocket Connector.....	16
Telegram BOT for Staff Alert.....	16
E-mail Sender for PiC (Person In Charge).....	17
Web Site.....	17
WebApp Backend.....	18
Student Notifications Data Flow.....	18
stakeholders involved in the system.....	18

Cardinality and typology of functionality available to the single stakeholder.....	19
Identified Technologies.....	19
<i>POC developed for the project.....</i>	<i>20</i>
<i>Time and Price.....</i>	<i>20</i>

INTRODUCTION

The present document is the main deliverable for the Erasmus+ KA2 - UNISAFE project which UNIVERSITÀ DI PAVIA is coordinating and taking part in ("UNIPV" from here on), it features the planning of the functional process, the feasibility study and the technology selection serviceable to the project.

The objective of this document is to formally assess the goals, the interfaces and peculiarities of the target system. As anticipated in the preliminary phase of the project, some feasibility tests or POC (proofs of concept) that have proven effective and functional to the end goal will be reusable in the first steps of the designing and implementing process.

DEFINITION OF EMERGENCY

For the project's objectives, the term "emergency" applies to any of the following situations and/or occurrences:

- Natural disasters (e.g. floodings, earthquakes, tsunamis, etc.)
- War incidents;
- Terrorism incidents;
- Mass shootings;
- Involvement into illegal activities.

PRESENT HANDLING OF EMERGENCIES

There are two kind of students managed by UNIPV:

- Outgoing students: from UniPV -> abroad
- Incoming students: abroad -> UniPV

The following two sections describe the current emergency handling for these groups of students.

OUTGOING STUDENTS: FROM UNIPV TO ABROAD

The International Mobility Unit of the University of Pavia is in charge of the management of the entire mobility process. It takes care of the students' selection, the preparation of mobility from the administrative point of view and supports the students while they are abroad. The idea of "support" includes the one to be provided in case of emergency.

Currently in UniPV there is not any system to continuously monitor emergency events worldwide. They strongly advise the students to register their stay in the Foreign Affairs Ministry portal "Viaggiare sicuri". However, UniPV cannot have any confirmation of their registration. So basically, the International mobility unit staff learns about incidents by accidental news reading activities. As soon as one of the members gets the information, they inform the other colleagues through the International Mobility Unit WhatsApp group.

Once the UniPV staff learn about the city, area or region where the emergency has taken place, they apply a filter per country in the management Excel file they maintain in order to identify the students who may potentially be involved in the event.

At this point, UniPV tries to get in touch with the identified students, basically only via e-mail. If they do not answer within a few hours, they try to get in contact via phone calls.

Most of the time UniPV obtains an “OK” from the students, so they are not required to provide further support, but in the unfortunate case where the mobile participants got involved in something major, they try to get in contact with them. If this attempt fails, the UniPV staff reaches out to the emergency contact provided by the student in the form filled in on arrival. At the same time, contact is established with the student’s host university.

During an emergency, it can also happen that the university International Relations office is contacted by the International Relations office of the host institution which wants to inform about the current status of the situation on campus and about the measures put in action in order to make sure that its incoming participants (including the ones from UNIPV) are safe and have been not affected by the emergency.

INCOMING STUDENTS: FROM ABROAD TO UNIPV

With regards to the Incoming students, currently the UniPV management system provides mostly communication support with the home Institution.

As soon as they learn about an emergency situation that involves one of the incoming students, immediately get in contact with the responsible people at the home institution from a prompt information delivery.

PROJECT’S NEEDS AND GOAL

The previous sections highlight a lack of an active monitoring system to handle emergencies on a global scale, this prevents an analytical, fast and efficient management of the process of contacting students. Much of the work required is manual and effectiveness could greatly benefit from the implementation of a modern, dedicated system, capable of near-real-time alerts for emergencies of different kinds occurring in several places around the world. The system would be able to provide the necessary automation to handle the students contacting phase employing a variety of technologies and means. The UniSafe project is geared towards improving and streamlining the current process through a number of custom-built software components. The team studied several processes, technologies and methodologies to identify the necessary tools to build a system for an efficient, shared and easy to handle project. The present document represents the system’s feasibility study and will illustrate its various steps.

FINAL OUTPUTS

The document was drafted at the end of the analysis process and the feasibility study, along with some proofs of concept:

The final document features:

- Data flow analysis
- Identification of stakeholders involved in the system
- Cardinality and types of functionality available to the single stakeholder
- Development time
- Identified technologies

- Operating systems, programming languages, databases, frameworks e and other pieces of software or libraries implemented by the finished system
- Description of ant POC developed for critical aspects of the system (e.g. lexical analyzer, Telegram bot)

In addition to the document, the finished product will contain a series of POC that will be developed during the feasibility analysis. As previously mentioned, these POC will be the technical starting point for developing the actual system.

MAJOR STEPS OF THE FEASIBILITY STUDY

The feasibility study was structured in function of the following major steps:

- Analysis of the general necessities
- Macroanalysis of the present situation
- Detailed analysis of the currently employed process
- Development of the main components
 - Registration data management
 - 24/7 World-wide monitoring
 - Semantic analysis (machine learning)
 - Interfacing with institutes of seismology
 - Alerting
 - SCR (Safety Check Request) multichannel communication
 - SCR answer collecting
 - Web application for managing and checking the process

GENERAL ANALYSIS OF THE PROCESS

In this step the team identified all the users involved with the system (stakeholders) and how they will interact with one another. A non-comprehensive list of stakeholders includes: students, local campus staff, system coordination staff, technical support.

DETAILED ANALYSIS OF THE PROCESS

Following an iterative approach this step will examine the responsibilities and interactions of the different users. Furthermore, all the technical and informational tools each user needs at their disposal to achieve their business objective will be defined (e.g. “a student needs to able to communicate their status to the university” or “The university needs to able to ask students abroad for information” and so on).

MULTICHANNEL COMMUNICATION

Both the university and the students need to able to send and receive alerts through a plethora of means. These means are not just a different method of exchanging information but each is characterized by different restrictions, strengths and accessibility. In this way it is

possible to compensate for malfunctions affecting any of them by relying on the others (e.g. “In the absence of data connection an SMS might still be delivered” or “If the student hasn’t installed Telegram they could get an alert via e-mail” and so forth). At any given moment, in the event of a “status request”, it will be possible to monitor students who have not answered yet, students who have already answered and how many have sent help requests.

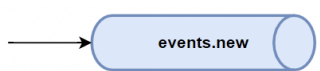
FEASIBILITY STUDY

Once all technical and technological tools have been defined for every stakeholder in the system, feasibility tests will need to be conducted to establish user-friendliness, effectiveness and efficiency.

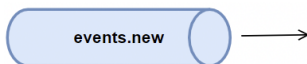
TECHNOLOGY SELECTION

By the end of the feasibility study, and in function of it, all the involved technological fields were identified. For each field the specific technology better suited to UNIPV and the system at large was selected.

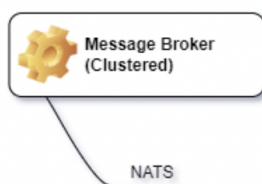
Here are the keys for understanding the tables below:



Message incoming into a message queue¹



Message from a message queue being read



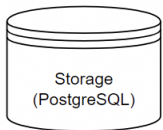
All the message queues are supplied by a component called Message Broker. Even though there are no strict requirements, at the time of this document’s drafting the selected Message Brokers is NATS with JetStream².

¹ A message queue is similar to the concept of an e-mail inbox. Sending a message to a queue means “sending an e-mail to a recipient” while reading a message from a queue means “reading a message someone sent from an inbox”.

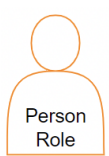
² <https://nats.io/>



Specialized component that will be developed as part of the software project. It will be presented with information on the implementing technology and a general description.

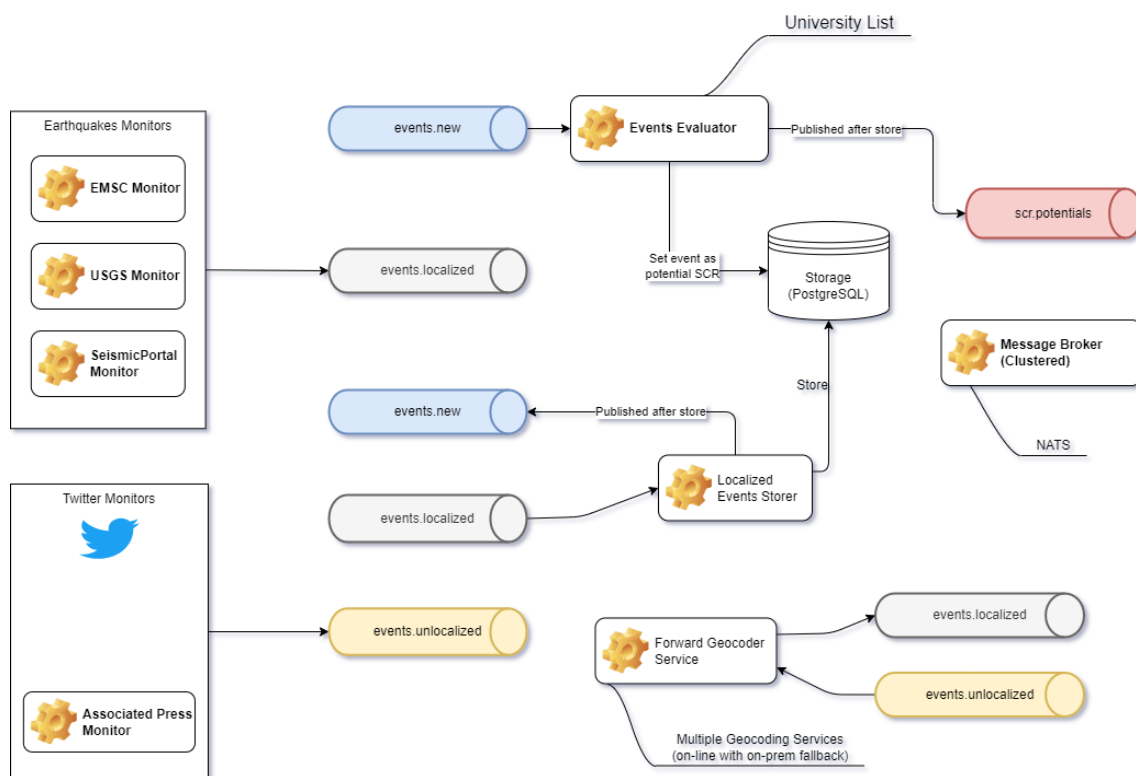


Relational database. Third party software component the project will make use of, it will not be developed from scratch. It requires effort for installation, training and configuration, but for all intents and purposes it is a standard tool available on the market.



It represents a role in the system. It does not refer to a single person but to a group of people who have a set role or, more broadly, handle specific duties.

MACRO COMPONENT 1: MONITORING



bit Time Professionals

Author: Daniele Teti <d.teti@bittime.it>

Figure 1: MONITORING BLOCK DIAGRAM

MACRO COMPONENT 2: HUMAN SCR EVALUATION

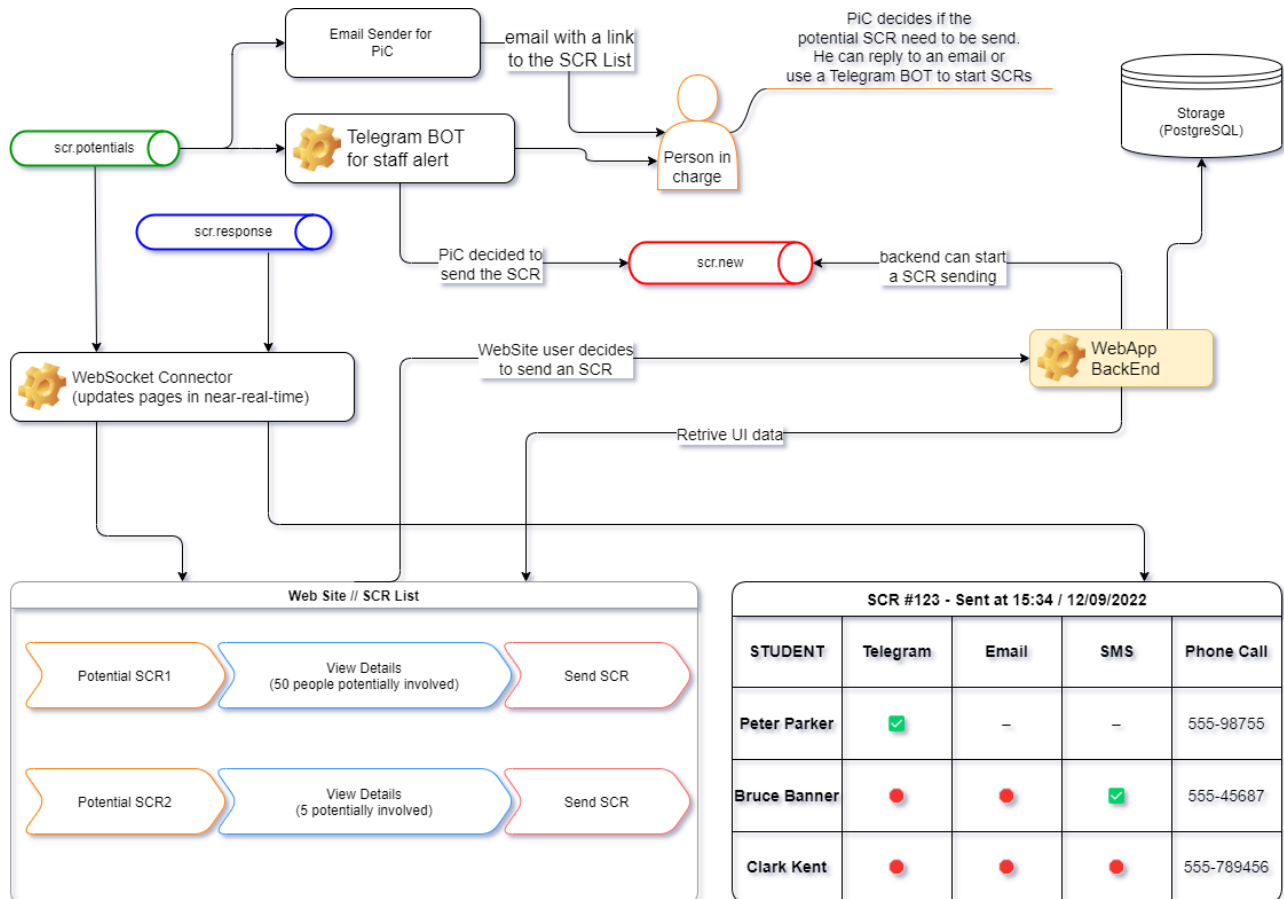
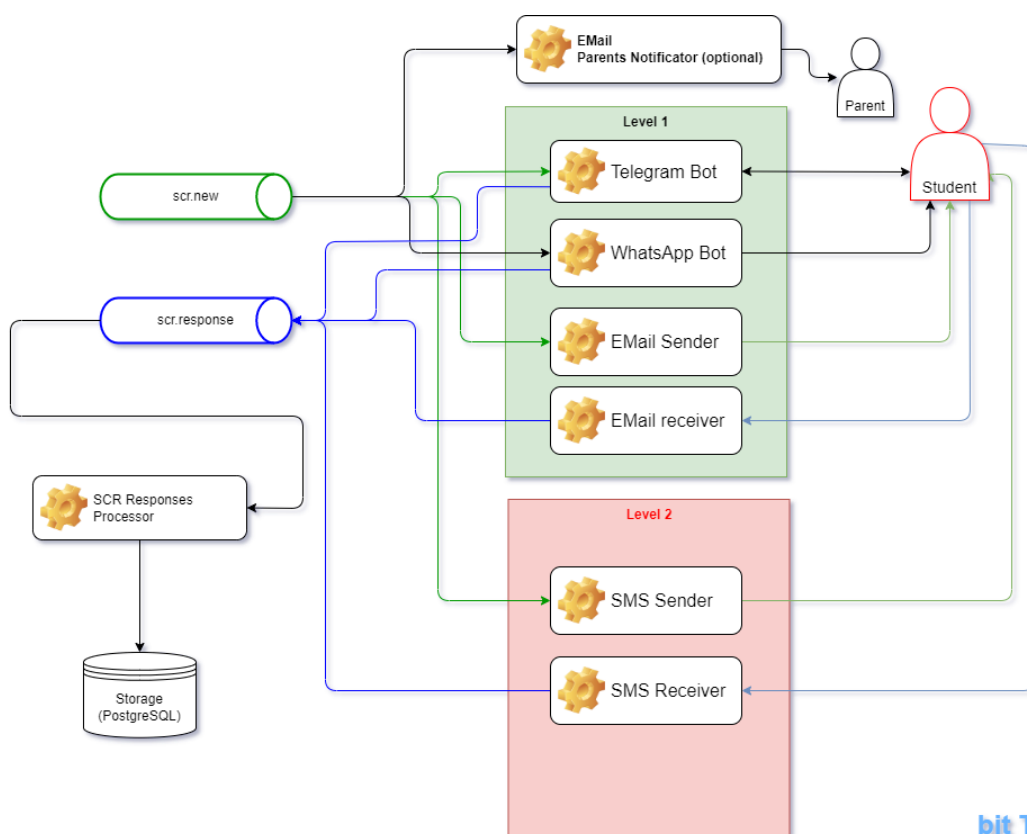


Figure 2: Human Safety Check Request (SCR) Evaluation BLOCK DIAGRAM

MACRO COMPONENT 3: NOTIFICATION



bit Time Professionals

Author: Daniele Teti <d.teti@bittime.it>

Figure 3: Students and (optional) parents notification BLOCK DIAGRAM

FEASIBILITY STUDY

This section will illustrate the outcomes and guidelines resulting from the feasibility study.

DATA FLOW ANALYSIS

The system is made of several data streams. For the purposes of the feasibility studies the possibility of accessing students and universities registration data was taken for granted.

The students' data in particular will have to contain the following information: registration number, name, family name, date of birth, date of departure, date of return, phone number, e-mail, host university.

The registration data of the universities will have to contain the following information: university name, state, province, region, latitude (approx.), longitude (approx.). In the event of physically detached faculties and/or departments every department will need to specify the required information. The granularity of the risk zone identification engine will regard as a "risk zone" even neighboring areas (in a radius of a few KM) to the affected one.

DATA FLOW MONITORING PHASE

The monitoring of information regarding emergencies is entrusted to two groups of components highlighted in Figure 1, like "Earthquakes Monitor" and "Twitter Monitor".

Earthquakes Monitor: Monitoring of news linked to earthquakes through the main dedicated websites. At the moment 3 websites have been identified, although the architecture allows the seamless integration of new ones or the replacement of the present ones. The currently selected websites are:

- EMSC (<https://www.emsc-csem.org/>)
 - From the website: "The European-Mediterranean Seismological Centre (EMSC) was founded in 1975, following a recommendation from the European Seismological Commission (ESC). The ESC is a regional commission of the International Association of Seismology and Physics of the Earth's Interior (IASPEI), itself a specialized association of the International Union of Geodesy and Geophysics (IUGG)."
- SeismicPortal (<https://www.seismicportal.eu/>)
 - This portal is driven and managed by the data collected by the EMSC. It provides a high-level interface to get the earthquakes data in near-real-time.
 - A proof of concept software has been developed for this source of information. It is a Python script able to get information from the APIs, retrieve latitude and longitude, compare the earthquake location to a university location and raise an alert if the Euclidean distance (https://en.wikipedia.org/wiki/Euclidean_distance) between the earthquake and

the university is below a threshold value.

- The example script can be found in the folder poc\monitors\earthquakes\seismicportal_test.py and this is an utilization example using test data with a very large radius of alert.

```
PS C:\DEV\unisafe_study\poc\monitors\earthquakes> python .\seismicportal_test.py 2
*****
** Showing worldwide events with magnitude >= 2.0 **
*****
[EVENT #1]
TIME: 2023-01-18T04:21:59.0Z
REGION: CENTRAL MEDITERRANEAN SEA (lat: 34.96, lon: 14.12)
MAGNITUDE: 5.0
DISTANCE: 380.52 Km
ALERT FOR: MESSINA UNIVERSITY
```

- USGS (<https://www.usgs.gov/>)
 - From the website: “Created by an act of Congress in 1879, the U.S. Geological Survey has evolved over the decades, matching its talent and knowledge to the progress of science and technology. The USGS is the sole science agency for the Department of the Interior. It is sought out by thousands of partners and customers for its natural science expertise and its vast earth and biological data holdings.”

EARTHQUAKES MONITOR

The “Earthquakes Monitor” sends messages when it detects seismic events on a global scale. These are highly localized events because earthquake notifications always include coordinates data. Messages of this kind are stacked in the “events.localized” queue.

TWITTER MONITORS

The “Twitter Monitors” block sends broadly localized messages (possibly containing just the name of the city where the event took place), before being sent the messages also undergo semantic analysis to ensure they contain information relevant to the system. Every piece of news is lexically and semantically analyzed with the purpose on highlighting those about earthquakes, tsunamis, destructive natural events and generally dangerous occurrences. Once relevant messages have been detected, but not accurately geographically pinpointed, they are sent to the “events.unlocalized” queue.

FORWARD GEOCODER SERVICE

The Forward Geocoder Service (FGS) is meant to increase the geographical accuracy of an event adding latitude and longitude to the ones that haven’t been geotagged. Let’s take an event like the following as an example (real tweet posted by the *The Associated Press* account):



BREAKING: Paris prosecutor: 89 victims killed in attack on Bataclan concert hall alone.

[Traduci il Tweet](#)

7:27 PM · 14 nov 2015

The FGS is responsible for adding geodata to the text “BREAKING: Paris prosecutor: 89 victims killed in attack on Bataclan concert hall alone”. In some cases, adding geospatial data is relatively easy and the data itself could be really accurate, in other cases it could prove challenging (vague data) or outright impossible (the text only mentions broad areas instead of exact locations).

After adding in location data, when possible, the FGS will place the message in the “events.localized” queue. From this point forward two kinds of events follow the same logic flow (geotagged and non-geotagged events).

LOCALIZED EVENTS STORER

Every geotagged event is backed by the Localize Events Storer (LES), before undergoing further evaluation. The backup is ensured by a PostgreSQL database cluster (refer to next paragraph). After being backed the message is added to the “events.new” queue ready to be evaluated by the “Events Evaluator”.

POSTGRESQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 35 years of active development on the core platform.

PostgreSQL is released under the PostgreSQL License³, a liberal Open Source license, similar to the BSD or MIT licenses.

PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organizations.

PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible. For example, you can define your own data types, build out custom functions, even write code from different programming languages without recompiling your database!

PostgreSQL tries to conform with the SQL standard where such conformance does not contradict traditional features or could lead to poor architectural decisions. Many of the features required by the SQL standard are supported, though sometimes with slightly differing syntax or function. Further moves towards conformance can be expected over time. As of the version 15 release in October 2022, PostgreSQL conforms to at least 170 of the 179

³ <https://opensource.org/licenses/postgresql>

mandatory features for SQL:2016 Core conformance. As of this writing, no relational database meets full conformance with this standard.

POSTGRESQL INEXHAUSTIVE FEATURES LIST

Below is an inexhaustive list of various features found in PostgreSQL, with more being added in every major release:

Data Types

- Primitives: Integer, Numeric, String, Boolean
- Structured: Date/Time, Array, Range / Multirange, UUID
- Document: JSON/JSONB, XML, Key-value (Hstore)
- Geometry: Point, Line, Circle, Polygon
- Customizations: Composite, Custom Types

Data Integrity

- UNIQUE, NOT NULL
- Primary Keys
- Foreign Keys
- Exclusion Constraints
- Explicit Locks, Advisory Locks

Concurrency, Performance

- Indexing: B-tree, Multicolumn, Expressions, Partial
- Advanced Indexing: GiST, SP-Gist, KNN Gist, GIN, BRIN, Covering indexes, Bloom filters
- Sophisticated query planner / optimizer, index-only scans, multicolumn statistics
- Transactions, Nested Transactions (via savepoints)
- Multi-Version concurrency Control (MVCC)
- Parallelization of read queries and building B-tree indexes
- Table partitioning
- All transaction isolation levels defined in the SQL standard, including Serializable
- Just-in-time (JIT) compilation of expressions

Reliability, Disaster Recovery

- Write-ahead Logging (WAL)
- Replication: Asynchronous, Synchronous, Logical

- Point-in-time-recovery (PITR), active standbys
- Tablespace

Security

- Authentication: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate, and more
- Robust access-control system
- Column and row-level security
- Multi-factor authentication with certificates and an additional method

Extensibility

- Stored functions and procedures
- Procedural Languages: PL/pgSQL, Perl, Python, and Tcl. There are other languages available through extensions, e.g. Java, JavaScript (V8), R, Lua, and Rust
- SQL/JSON path expressions
- Foreign data wrappers: connect to other databases or streams with a standard SQL interface
- Customizable storage interface for tables
- Many extensions that provide additional functionality, including PostGIS

Internationalization, Text Search

- Support for international character sets, e.g. through ICU collations
- Case-insensitive and accent-insensitive collations
- Full-text search

PostgreSQL has been proven to be highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL clusters in production environments that manage many terabytes of data, and specialized systems that manage petabytes.

MESSAGE BROKER - NATS

NATS is a connective technology built for the ever increasingly hyper-connected world. It is a single technology that enables applications to securely communicate across any combination of cloud vendors, on-premise, edge, web and mobile, and devices. NATS consists of a family of open source products that are tightly integrated but can be deployed easily and independently. NATS is being used globally by thousands of companies, spanning use-cases including microservices, edge computing, mobile, IoT and can be used to augment or replace traditional messaging.

The NATS Server acts as a central nervous system for building distributed applications. Client APIs are provided in over 40 languages and frameworks including Go, Java, JavaScript/TypeScript, Python, Ruby, Rust, C#, C, and NGINX. Real time data streaming, highly resilient data storage and flexible data

retrieval are supported through JetStream, the next generation streaming platform built into the NATS server. Check out the full list of NATS clients.

NATS was created in response to the market need for a simple, secure, and connective technology. NATS is currently deployed in some of the largest cloud platforms, including: VMware, CloudFoundry, Baidu, Siemens, and GE. NATS is 100% free to use under the Apache-2.0 Open Source License. NATS is unique in its simplicity and performance, and as a result powers some of the largest production environments.

EVENTS EVALUATOR

The Event Evaluator is the component in charge of evaluating whether an event from the “events.new” queue is relevant for the purposes of the system. According to the geotagging of universities and relative attending student the system evaluates if the location of an event should be kept under control. The Event Evaluator uses a set of criteria to give more or less weight to the events it has to evaluate depending on the event’s type. In any case the event to be evaluated is updated on the storage to show its attention level. In case the attention level exceeds a preset threshold the event is also added to the “scr.potentials” queue, as well as being saved on the storage. This queue is the entry point of the next figure: Human SCR Evaluation.

HUMAN SCR EVALUATION PHASE DATA FLOW

The macro component “Human SCR Evaluator” receives data from its predecessor (Monitoring) through the “scr.potentials” queue. This queue contains all the potential risk events that could affect travelling students. This queue is used by the “WebSocket Connector” component, by the “Telegram BOT for Staff Alert” and by the “E-mail Sender for PiC” as described in the following paragraphs.

WEBSOCKET CONNECTOR

When the “scr.potentials” queue receives a message, whoever is using the web app is immediately alerted of the potential threat by web interface itself and will be able to confirm the SCR or not. The component channeling all messages from “scr.potentials” to the web interface is the “WebSocket Connector”. In case an SCR is confirmed a message is added to the “scr.new” queue.

TELEGRAM BOT FOR STAFF ALERT

The person in charge of evaluating the potential SCR is always alerted both via web application through the “WebSocket Connector” and through the “Telegram BOT for Staff Alert”. This channel for notifications is especially important for alerts that could reach the person in charge on a public holiday or at night, times when the person is not physically using the web application. The message sent via Telegram will report the potential SCR identification data, the kind of event and a link or text to learn more about it. The person in charge will be able to confirm the potential SCR launching the SCR delivery process directly from the Telegram app. The message is sent to everyone flagged as “person in charge” at that moment. In case the SCR is confirmed a message will be added to the “scr.new” queue.

E-MAIL SENDER FOR PiC (PERSON IN CHARGE)

The communication channel “E-mail Sender For PiC” represents an alternative to the message sent by the “Telegram Bot For Staff Alert” and grants the same functionality though via e-mail (potential SCR identification, source checking, SCR confirmation). This variety is functional and necessary given the urgency and inherent in this kind of communication. The e-mail is sent to those in the group flagged as “Person in charge”. In case the SCR is confirmed a message will be added to the “scr.new” queue.

WEB SITE




The web application offers all the functionalities the users need to manage registration data, potential SCR monitor, actually sent SCR and a list of all the answers sent in by the students. During the feasibility study the team focused on the user interaction in the SCR evaluation and monitoring phases. It will be possible to look up the SCR history (see dedicated paragraph) and the trend of the students’ answers to a specific SCR.

#SCR	Proposed At	Type	Event Location	Universities Involved	Students Involved	Confirmed At	Not Confirmed At
1234	12/10/2023 12:34	Eartquake	New York	NYU	32	12/10/2023 12:39	-
1235	01/12/2023 18:05	Terrorism	Jakarta	Jakarta University	4	01/12/2023 18:10	-
1236	20/01/2024 23:36	Eartquake	Catania	Università di Catania	10	-	21/01/2024 00:48
1237	11/03/2024 05:07	Eartquake	Rome	La Sapienza	23	11/03/2024 05:13	-
1238	30/04/2024 10:38	Flood	Berlin	Freie Universität Berlin	10	30/04/2024 10:44	-
1239	19/06/2024 16:10	Tsunami	Fukushima	Fukushima University	45	19/06/2024 16:15	-
1240	08/08/2024 21:41	Flood	Paris	Université Paris Cité	32	-	08/08/2024 22:53
1241	28/09/2024 03:12	Tsunami	Haiti	State University of Haiti	2	-	28/09/2024 04:24

Figure 4: SCR History

This trend is represented with a grid that auto-updates upon recieving answers by the students.

The following image is just an example of which details can be displayed. Here is an explanation of the icons:

-  SCR sent, no answer for this channel yet
-  Positive answer (“I’m OK”)
-  Negative answer (“I’m in trouble”)

The multi-channel messages will be sent starting with the leftmost channel (Telegram) and will continue right. The message to the first two channels (Telegram and e-mail) will be sent at the same time. If no answer comes back after a set time limit (e.g. 5 minutes) all the remaining messages are sent.

SCR #1237 – ROME – La Sapienza (Sent at 15:34 – 2023-01-23)					
STUDENT	Telegram	Email	WhatsApp	SMS	Phone
Mario Rossi	✓	-	-	-	555-2345876
Debora Bianchi	🔔	✓	-	-	555-29091983
Mattia Bianchi	🔔	🔔	✓	-	555-17112011
Antonio Verdi	🔔	🔔	🔔	✗	555-9857734
Giorgio Giorgi	🔔	🔔	✗	-	555-8796345
Antonella Nella	🔔	🔔	✓	-	555-897876
Giacomo Limone	✓	-	-	-	555-25543243
Giovanni Vino	🔔	🔔	🔔	✓	555-04111979

Figure 5: SCR Notification Replies

To avoid confusion, after the first answer by a student, from any of the channels, the remaining messages for that SCR won't be sent anymore.

WEBAPP BACKEND

The WebApp backend grants the necessary API to the WebSite and makes it possible to launch the process for sending the SCR. It communicates directly with the storage (PostgreSQL) managing registration and operational data.

STUDENT NOTIFICATIONS DATA FLOW

The macro component "Student Notifications" receives data from its predecessor (Human SCR Evaluator) through the "scr.new" and adds the reply messages to the "scr.response" queue.

When a new message is added to the "scr.new" queue messages are immediately sent to the "Level 1" channels:

- Telegram
- WhatsApp
- E-mail

It is also possible to send an e-mail to the student's emergency contact (a parent or a guardian). The e-mail to the emergency contact serves to communicate that the university has taken action to ask the student for information on their status and about the emergency situation.

If after a set time limit (e.g. 5 minutes) no answer came through the Level 1 channels, messages to Level 2 channels will be sent: SMS.

All the answers, of every kind, will be added to the "scr.response" queue and the same time will be backed on the Storage (PostgreSQL).

STAKEHOLDERS INVOLVED IN THE SYSTEM

- Students
- Parents / emergency contacts
- University Staff in charge of SCR assessment during work hours

- University Staff in charge of SCR assessment during non-work hours

CARDINALITY AND TYPOLOGY OF FUNCTIONALITY AVAILABLE TO THE SINGLE STAKEHOLDER

- Students
 - They complete the registration with one or more channel for receiving the SCR
- Parents
 - Will receive the alert (optional) that an SCR will be sent in the area their children are located in
- University staff in charge SCR assessment during work hours
 - They use the web system;
 - They assess whether to send the SCR;
 - They monitor the answers SCR;
 - They decide if the students who have not responded within the set limit need to be called;
- University Staff in charge of SCR assessment during non-work hours (availability)
 - They receive a message/ e-mail in which they can read the reasons for the alert and eventually send an SCR.

IDENTIFIED TECHNOLOGIES

The technologies identified at moment are:

- The system will work on Ubuntu Linux systems (or in any case Debian based);
- All the server-side software will be developed in Python and will implement several open-source code libraries;
- The web client will be developed in ReactJS and supporting technologies;
- The ODBMS will be PostgreSQL;
- The message broker will be NATS;
- The Geocoding engine will be considered in the implementation stage. An open source engine⁴ will be chosen, if not available, or not powerful enough, it will be necessary to employ a commercial Geocoding engine. For reference purposes the following figure contains the rates for the Google Maps Geocoding service.

MONTHLY VOLUME RANGE (Price per REQUEST)		
0-100,000	100,001-500,000	500,000+
0.005 USD per each (5.00 USD per 1000)	0.004 USD per each (4.00 USD per 1000)	Contact Sales for volume pricing

The monthly number of requests will surely be below 100,000.

⁴ <https://github.com/osm-search/Nominatim>

POC Developed For The Project

During the feasibility study 2 POCs were produced:

- Telegram BOT
- Earthquakes geotagging and risk assessment

The source code for the POC is attached to the supplementary material for the present document.

Time And Price

The project can be delivered in all its parts after 140 man days. The price for system engineering, development, testing and deployment is €214,369.00 plus VAT.

Anything that is not explicitly mentioned in this document will not be included in the final system.