UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

LAB. SOFTWARE AVANZADO

AUX. DIEGO MOLINA

DOCUMENTACIÓN FASE 1

JONATHAN MARCOS VALIENTE GONZÁLEZ

201931581

5 DE MARZO DE 2024, GUATEMALA

ÍNDICE

DOCUMENTACIÓN DE REQUISITOS

REQUISITOS

Requerimientos Funcionales

- El sistema debe contar con un apartado de creación de tickets, dónde el usuario especifica parámetros como datos de contacto, archivos de evidencia, descripción del problema y tipo de problema.
- El sistema cuenta con un apartado para buscar los tickets pendientes en base a los datos de contacto del cliente.
- El usuario tiene permitido ver los tickets que le pertenecen, rastrear el estado del ticket y mandar mensajes al agente que está resolviendo ese ticket.
- Un agente puede rastrear tickets, asignarse el ticket, cambiar el estado del ticket, cambiar la prioridad del ticket, resolver y chatear con el usuario del ticket.
- El sistema cuenta con registro de usuarios agentes y administradores, dónde solo los administradores pueden agregar a otros usuarios agentes o administradores.
- Los administradores pueden ver informes acerca de la resolución de tickets, importar a una herramienta de CRM por medio de un csv.
- Tras la resolución de un ticket el cliente llena una encuesta de retroalimentación.

Requerimientos no Funcionales

- El sistema debe funcionar con herramientas en la nube de Google.
- Manejar seguridad web en el manejo de los usuarios agente y administrador.
- El sistema debe ser escalable.

FUNCIONALIDADES

Creación de Tickets

```
@PostMapping("/create-ticket")
public ResponseEntity<?> createTicket(@Validated @RequestBody String json){
   try{
       JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
       //TicketType ticketType =
ticketTypeService.getByType(jsonNode.get("ticketType").asText());
       TicketType ticketType =
ticketTypeService.findById(jsonNode.get("ticketType").asInt());
       //TicketPriority ticketPriority =
ticketPriorityService.getByPriority(jsonNode.get("ticketPriority").asText());
       TicketPriority ticketPriority =
ticketPriorityService.findById(jsonNode.get("ticketPriority").asInt());
       // Obtener la lista de URLs del campo "urls"
       JsonNode urlsNode = jsonNode.get("files");
       List<String> urlsList = new ArrayList<>();
       for (JsonNode urlNode : urlsNode) {
           urlsList.add(urlNode.asText());
       }

       System.out.println("createTicket:"+ticketType.getType());
       Ticket ticket = new Ticket(
               jsonNode.get("email").asText(),
               jsonNode.get("name").asText(),
               jsonNode.get("lastName").asText(),
               jsonNode.get("phone").asText(),
               jsonNode.get("description").asText(),
               ticketType,
               ticketPriority,
               null
       );
       //Creacion del ticket
       ticket = ticketService.createTicket(ticket);
       //Create Elements
       for(String url:urlsList){
           ticketElementService.saveElement(ticket,url);
       }
       //Creacion del tracking
       State_of_Ticket stateOfTicket =
stateOfTicketService.getReferencedById(1);
       TicketTracking ticketTracking =
ticketTrackingService.createTrack(ticket,stateOfTicket);

       //Creacion del historial
       historyOfCommunicationService.saveLog(
               ticketTracking,
               "Se creo el ticket de
tipo:"+ticket.getTicketType().getType()+", de
prioridad:"+ticket.getPriority().getPriority()
       );

       return ResponseEntity.status(HttpStatus.CREATED).build();
   }catch(Exception ex){
       ex.printStackTrace();
       System.out.println("Error: "+ex.getMessage());
       return new ResponseEntity<>(new Message("Error en creacion de tickets:
"+ex.getMessage()), HttpStatus.BAD_REQUEST);
   }
}
```

Seguimiento de Tickets

```
@GetMapping("/getTickets")
```

```java
public ResponseEntity<?> getTickets(@Validated @RequestParam("email") String
email){
   try{
       List<Map<String,Object>> listTokens =
ticketService.getTicketsEmail(email);
       List<Map<String, Object>> modifiedList = new ArrayList<>();
       for (Map<String, Object> map : listTokens) {
           Map<String, Object> modifiedMap = new HashMap<>(map); // Crear una
copia modificable del mapa actual
           String ticket = modifiedMap.get("ticketNumber").toString();
           int ticketNumber = Integer.parseInt(ticket);
           List<String> elements =
ticketService.getTicketsElements(ticketNumber);
           modifiedMap.put("files", elements);
           modifiedList.add(modifiedMap); // Agregar el mapa modificado a la
nueva lista
       }

       Map<String, Object> provider = new HashMap<>();
       provider.put("tickets", modifiedList);
       return new ResponseEntity<>(provider, HttpStatus.OK);
   }catch(Exception ex){
       ex.printStackTrace();
       return new ResponseEntity<>(new Message("Error obtencion de tickets:
"+ex.getMessage()), HttpStatus.BAD_REQUEST);
   }
}
```

Resolución de Tickets

```java
@PostMapping("/change-priority")
public ResponseEntity<?> changeTicketPriority(@Validated @RequestBody String
json){
   try{
       JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
       // Find the new Priority
       TicketPriority priority = ticketPriorityService.getByPriority(
               jsonNode.get("priority").asText()
       );
       //Find the ticket
       Ticket ticket = ticketService.getReferenceById(
               jsonNode.get("ticketNumber").asInt()
       );

       ticket.setPriority(priority);
       ticketService.createTicket(ticket);

       TicketTracking ticketTracking = ticketTrackingService.findById(
               jsonNode.get("ticketNumber").asInt()
       );
       ticketTracking.setDateLastUpdation(new Date());
       ticketTracking = ticketTrackingService.saveTicket(ticketTracking);
       historyOfCommunicationService.saveLog(
               ticketTracking,
               "Se cambio la prioridad del ticket a
"+jsonNode.get("priority").asText()
       );
       return ResponseEntity.status(HttpStatus.CREATED).build();
   }catch(Exception ex){
       return new ResponseEntity<>(new Message("Error en cambio de prioridad
en ticket: "+ex.getMessage()), HttpStatus.BAD_REQUEST);
   }
}
@PostMapping("/change-state")
public ResponseEntity<?> changeTicketState(@Validated @RequestBody String
json,
                                          @RequestHeader("Authorization")
String authorizationHeader){
   try{
       if(!jwtChecker.initCheck(authorizationHeader)){
           return new ResponseEntity<>(new Message("Sesion invalida"),
HttpStatus.FORBIDDEN);
       }
       JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
       // Find the new Priority
       State_of_Ticket stateOfTicket = stateOfTicketService.getByState(
               jsonNode.get("state").asText()
       );
       //Find the ticket
       TicketTracking ticketTracking = ticketTrackingService.findById(
               jsonNode.get("ticketNumber").asInt()
       );
       ticketTracking.setState(stateOfTicket);
       ticketTracking.setDateLastUpdation(new Date());
       ticketTracking=ticketTrackingService.saveTicket(ticketTracking);
       historyOfCommunicationService.saveLog(
               ticketTracking,
               "Se cambio el estado del ticket a
"+(jsonNode.get("state").asText())
       );
       return ResponseEntity.status(HttpStatus.OK).build();
   }catch(Exception ex){
       return new ResponseEntity<>(new Message("Error en cambio de estado del
ticket: "+ex.getMessage()), HttpStatus.BAD_REQUEST);
   }
}

@PostMapping("/solve")
public ResponseEntity<?> solve(@Validated @RequestBody String json){
   try{
       JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
       TicketTracking ticketTracking = ticketTrackingService.findById(
               jsonNode.get("ticketNumber").asInt()
       );
       ticketTracking.setProblemSolved(true);
```

```
        ticketTracking.setDateLastUpdation(new Date());
        ticketTracking=ticketTrackingService.saveTicket(ticketTracking);
        historyOfCommunicationService.saveLog(
                ticketTracking,
                "Se resolvio el ticket"
        );
        return ResponseEntity.status(HttpStatus.OK).build();
    }catch(Exception ex){
        return new ResponseEntity<>(new Message("Error al resolver el ticket: 
"+ex.getMessage()), HttpStatus.BAD_REQUEST);
    }
}

@PostMapping("/trackLogs")
public ResponseEntity<?> getLogs(@Validated @RequestBody String json){
    try{
        JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
        int ticketNumber = jsonNode.get("ticketNumber").asInt();
        List<History_of_CommunicationDTO> list = 
historyOfCommunicationService.getLogs(ticketNumber);
        return new ResponseEntity<>(list, HttpStatus.OK);
    }catch(Exception ex){
        return new ResponseEntity<>(new Message("Error, no se pudieron obtener 
los logs: "+ex.getMessage()), HttpStatus.BAD_REQUEST);
    }
}

@PostMapping("/assign-agent")
public ResponseEntity<?> assignAgent(@Validated @RequestBody String json,
                                    @RequestHeader("Authorization") String 
authorizationHeader){
    try{
        if (!jwtChecker.initCheck(authorizationHeader)) {
            // Registra una sesión inválida
            return new ResponseEntity<>(new Message("Sesión inválida"), 
HttpStatus.FORBIDDEN);
        }
        JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
        String agentUsername = jwtChecker.getSubject(authorizationHeader);
        User agent = userService.getUser(agentUsername);
        TicketTracking ticketTracking = 
ticketTrackingService.findById(jsonNode.get("ticketNumber").asInt());
        ticketTracking.setAgent(agent);
        ticketTracking = ticketTrackingService.saveTicket(ticketTracking);
        historyOfCommunicationService.saveLog(
                ticketTracking,
                "El ticket se asigno a "+ticketTracking.getAgent().getName()+
                        " "+ticketTracking.getAgent().getLastName()+" usuario: 
"+ticketTracking.getAgent().getUsername()
        );
        return ResponseEntity.status(HttpStatus.CREATED).build();
    }catch(Exception ex){
        return new ResponseEntity<>(new Message("Error en cambio de agente en
ticket: "+ex.getMessage()), HttpStatus.BAD_REQUEST);
    }
}

@PostMapping("/visualize")
public ResponseEntity<?> visualize(@Validated @RequestBody String json){
    try{
        JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
        int ticketNumber = jsonNode.get("ticketNumber").asInt();
        TicketTracking ticketTracking =
ticketTrackingService.findById(ticketNumber);
        return new ResponseEntity<>(ticketTracking, HttpStatus.OK);
    }catch(Exception ex){
        return new ResponseEntity<>(new Message("Error en obtencion de datos
del cliente "+ex.getMessage()), HttpStatus.BAD_REQUEST);
    }
}

@PostMapping("/visualizeTicket")
public ResponseEntity<?> viewTicket(@Validated @RequestBody String json,
                                    @RequestHeader("Authorization") String
authorizationHeader){
    try{
        //Validacion del JWT
        jwtChecker.checkJWT(authorizationHeader);

        JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
        int ticketNumber = jsonNode.get("ticketNumber").asInt();
        //Obtener El track del ticket y el historial
        TicketTracking ticketTracking =
ticketTrackingService.findById(ticketNumber);
        List<History_of_CommunicationDTO> historyOfCommunication =
historyOfCommunicationService.getLogs(ticketNumber);

        //Armar la respuesta
        Map<String, Object> response = new HashMap<>();
        response.put("ticket", ticketTracking);
        response.put("history", historyOfCommunication);
        return new ResponseEntity<>(response, HttpStatus.OK);
    }catch(Exception ex){
        return new ResponseEntity<>(new Message("Error en obtencion de datos
del cliente "+ex.getMessage()), HttpStatus.BAD_REQUEST);
    }
}
```

Gestión de Usuarios

```
@PostMapping("/register")
public ResponseEntity<?> registerUser(@Validated @RequestBody String json) {
    try{
        return register(json);
```

```
   }catch(Exception ex){
       ex.printStackTrace();
       return new ResponseEntity<>(new Message("Error en registro:
"+ex.getMessage()), HttpStatus.BAD_REQUEST);
   }
}

/**
* Provide
* {
*      username:
*      password:
*      userType: [as Role or Type] [options: cliente, agente, administrador]
*      name:
*      lastName:
*      phone:
* }
* @param json
* @return
*/
@PostMapping("/registerSpecial")
public ResponseEntity<?> registerUserProtected(@Validated @RequestBody String
json,
                                              @RequestHeader("Authorization")
String authorizationHeader) {
   try{
       if (!jwtChecker.initCheck(authorizationHeader)) {
           // Registra una sesión inválida
           return new ResponseEntity<>(new Message("Sesión inválida"),
HttpStatus.FORBIDDEN);
       }

       return register(json);
   }catch(Exception ex){
       ex.printStackTrace();
       return new ResponseEntity<>(new Message("Error en registro:
"+ex.getMessage()), HttpStatus.BAD_REQUEST);
   }
}

private ResponseEntity<?> register(String json) throws TicketException,
NoSuchAlgorithmException {
   JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
   String passwordEncrypted =
encrypter.encrypt(jsonNode.get("password").asText());
   UserType userType =
userTypeService.getByType(jsonNode.get("userType").asText());

   if(userService.usernameExists(jsonNode.get("username").asText())){
       throw new TicketException("Usuario ya existe");
   }

   User user = new User(
           jsonNode.get("username").asText(),
           passwordEncrypted,
           jsonNode.get("name").asText(),
           jsonNode.get("lastName").asText(),
           jsonNode.get("phone").asText(),
           userType
   );
   //Registramos el usuario
   userService.registerUser(user);
   return ResponseEntity.status(HttpStatus.CREATED).build();
}

/**
* Provide:{
*     username: String
*     password: String
* }
*
* Returns: {
*     token: String
*     user_type: number
* }
* @param json
* @return
*/
@PostMapping("/login")
public ResponseEntity<?> login(@Validated @RequestBody String json){
   try{
       JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
       String username = jsonNode.get("username").asText();
       String passwordEncrypted =
encrypter.encrypt(jsonNode.get("password").asText());
       User user = userService.getUser(username);
       if(user.getUsername().equalsIgnoreCase(username) &&
       user.getPassword().equals(passwordEncrypted)){
           //handle login
           Map<String, Object> map = new HashMap<>();
           map.put("username", username);
           Map<String, Object> response = new HashMap<>();
           response.put("jwt", jwtProvider.generateToken(username, map));
           response.put("user_type", user.getUserType());
           return new ResponseEntity<>(response, HttpStatus.OK);
       }else{
           throw new TicketException("El usuario o la contraseña son
incorrectos");
       }
   }catch(Exception ex){
       return new ResponseEntity<>(new Message("Error en Inicio de sesion:
"+ex.getMessage()), HttpStatus.NOT_ACCEPTABLE);
   }
}

@PostMapping("/validate")
```

```
public ResponseEntity<?> validateJWT(@RequestHeader("Authorization") String
authorizationHeader){
   try{
       jwtChecker.checkJWT(authorizationHeader);
       System.out.println("JWT Valido");
       return new ResponseEntity<>(new Message("Autorizado"), HttpStatus.OK);
   }catch(Exception ex){
       System.out.println("Sesion invalida");
       return new ResponseEntity<>(new Message("Sesion invalida:
")+ex.getMessage(), HttpStatus.FORBIDDEN);
   }
}
```

Informes y Análisis

```
@PostMapping("/getSurveys")
public ResponseEntity<?> getSurveys(@RequestHeader("Authorization") String
authorizationHeader){
   try{
       jwtChecker.checkJWT(authorizationHeader);
       return new ResponseEntity<>(surveyService.getSurveys(),
HttpStatus.OK);
   }catch(Exception ex){
       return  new ResponseEntity<>(new Message("Error al obtener las
encuestas"), HttpStatus.BAD_REQUEST);
   }
}
```

Funcionalidades Adicionales

```
@PostMapping("/saveSurvey")
public ResponseEntity<?> responseServer(@Validated @RequestBody String json){
   try{
       JsonNode jsonNode = new JsonOptions().parseStringJsonNode(json);
       int ticketNumber = jsonNode.get("ticketNumber").asInt();
       Ticket ticket = ticketService.getReferenceById(ticketNumber);
       int satisfaction = jsonNode.get("satisfaction").asInt();
       int timeService = jsonNode.get("timeService").asInt();
       int qualityService = jsonNode.get("qualityService").asInt();
       Survey survey = new Survey(
               ticket,
               satisfaction,
               timeService,
               qualityService
       );
       surveyService.saveSurvey(survey);
       return ResponseEntity.status(HttpStatus.OK).build();
   }catch(Exception ex){
       return new ResponseEntity<>(new Message("Error en la respuesta de la
encuesta :"+ ex.getMessage()), HttpStatus.BAD_REQUEST);
```

```
    }
}

@PostMapping("/getCSV")
public ResponseEntity<?> getCSV(@RequestHeader("Authorization") String
authorizationHeader){
    try{
        jwtChecker.checkJWT(authorizationHeader);
        return new ResponseEntity<>(surveyService.getCsv(), HttpStatus.OK);
    }catch(Exception ex){
        return new ResponseEntity<>(new Message("Error en obtener el csv"),
HttpStatus.BAD_REQUEST);
    }
}
```

TIPOS DE USUARIOS Y PERMISOS

CLIENTE

- Crear tickets
- Rastrear Tickets
- Ver tutoriales
- Mensajear con agente atendiente del ticket
- Resolver encuestas de retroalimentación.

AGENTE

- Asignar agente a un ticket
- Resolver ticket
- Cambiar estado del ticket
- Cambiar prioridad del ticket
- Ver detalles del ticket
- Mensajear con cliente solicitante del ticket

ADMINISTRADOR

- Crear usuarios agente y administrador
- Generar informes de retroalimentación
- Generar archivos csv para conexión con herramientas CRM

CONTRATOS DE MICROSERVICIOS

Cloud Storage Service Level Agreement (SLA)

During the Term of the agreement under which Google has agreed to provide Google Cloud Platform to Customer (as applicable, the "Agreement"), the Covered Service will provide a Monthly Uptime Percentage to Customer as follows (the "Service Level Objective" or "SLO"):

Covered Service	**Monthly Uptime Percentage**
Standard storage class in a multi-region or dual-region location of Cloud Storage	>= 99.95%
Standard storage class in a regional location of Cloud Storage; Nearline, Coldline, or Archive storage class in a multi-region or dual-region location of Cloud Storage	>= 99.9%
Nearline, Coldline, or Archive storage class in a regional location of Cloud Storage; Durable Reduced Availability storage class in any location of Cloud Storage	>= 99.0%

The turbo replication feature will provide the following Service Level Objectives:

Any Cloud Storage bucket for which the turbo replication feature has been activated for the entire billing month	Monthly Replication Time Conformance >= 99.0% Monthly Replication Volume Conformance >= 99.9%

If Google does not meet the SLO, and if Customer meets its obligations under this SLA, Customer will be eligible to receive the Financial Credits described below. Monthly Uptime Percentage, Monthly Replication Time Conformance, Monthly Replication Volume Conformance, and Financial Credit are determined on a calendar month basis per Project or, with respect to the turbo replication feature, per Cloud Storage bucket. This SLA states Customer's sole and exclusive remedy for any failure by Google to meet the SLO. Capitalized terms used in this SLA, but not defined in this SLA, have the meaning set forth in the Agreement. If the Agreement authorizes the resale or supply of Google Cloud Platform under a Google Cloud partner or reseller program, then all references to Customer in this SLA mean Partner or Reseller (as applicable), and any Financial Credit(s) will only apply for impacted Partner or Reseller order(s) under the Agreement.

Definitions

The following definitions apply to the SLA:

- **"Back-off Requirements"** means, when an error occurs, the Application is responsible for waiting for a period of time before issuing another request. This means that after the first error, there is a minimum back-off interval of 1 second and for each consecutive error, the back-off interval increases exponentially up to 32 seconds.
- **"Bad RPO Minute"** means each minute where there are outstanding replications taking longer than 15 minutes for the turbo replication feature to complete.
- **"Covered Service"** means Cloud Storage.
- **"Error Rate"** means the number of Valid Requests that result in a response with an HTTP Status in the 500 range divided by the total number of Valid Requests during the applicable period, represented as a percentage. Repeated identical requests do not count towards the Error Rate unless they conform to the Back-off Requirements.
- **"Financial Credit"** means the following for the Standard storage class in a multi-region or dual-region location of Cloud Storage:

Monthly Uptime Percentage	Percentage of monthly bill for the Standard storage class in a multi-region or dual-region location of Cloud Storage that does not meet SLO that will be credited to Customer's future monthly bills
99.0% – < 99.95%	10%

95.0% – < 99.0%	25%
< 95.0%	50%

- **"Financial Credit"** means the following for the Standard storage class in a regional location of Cloud Storage, and the Nearline, Coldline, or Archive storage class in a multi-region or dual-region location of Cloud Storage:

Monthly Uptime Percentage	Percentage of monthly bill for the Standard storage class in a regional location of Cloud Storage or the Nearline, Coldline, or Archive storage class in a multi-region or dual-region location of Cloud Storage that does not meet SLO that will be credited to Customer's future monthly bills
99.0% – < 99.9%	10%
95.0% – < 99.0%	25%
< 95.0%	50%

- **"Financial Credit"** means the following for the Nearline, Coldline, or Archive storage class in a regional location of Cloud Storage, and Durable Reduced Availability storage class in any location of Cloud Storage:

Monthly Uptime Percentage	Percentage of monthly bill for the Nearline, Coldline, or Archive storage class in a regional location of Cloud Storage, or the Durable Reduced Availability storage class in any location of Cloud Storage that does not meet SLO that will be credited to Customer's future monthly bills
98.0% – < 99.0%	10%
95.0% – < 98.0%	25%
< 95.0%	50%

- **"Financial Credit"** means the following for a Cloud Storage bucket for which the turbo replication feature has been activated for the entire calendar month:

Monthly Replication Time Conformance	Monthly Replication Volume Conformance	Percentage of monthly bill for the Cloud Storage bucket that does not meet either or both SLO(s) that will be credited to Customer's future monthly bills*
98.0% – < 99.0%	99.0% – < 99.9%	10%
95.0% – < 98.0%	95.0% – < 99.0%	25%
<95.0%	< 95.0%	50%

*If both a Monthly Uptime Percentage SLO and turbo replication SLO are affected in a given month within the same Project, then Customer will only be entitled to the greater of the two corresponding Financial Credits (not both).

- **"Monthly Replication Time Conformance"** means 100% minus the percentage of Bad RPO Minutes during the calendar month.
- **"Monthly Replication Volume Conformance"** means 100% minus the percentage of applicable Cloud Storage objects in the calendar month that the turbo replication feature did not replicate within 15 minutes.
- **"Monthly Uptime Percentage"** means 100%, minus the average of Error Rates measured over each five minute period during the calendar month.
- **"Valid Requests"** are requests that conform to the Documentation, and that would normally result in a non-error response.

Customer Must Request Financial Credit

In order to receive any of the Financial Credits described above, Customer must notify Google technical support within 30 days from the time Customer becomes eligible to receive a Financial Credit. Failure to comply with this requirement will forfeit Customer's right to receive a Financial Credit.

Maximum Financial Credit

The aggregate maximum number of Financial Credits to be issued by Google to Customer in a single billing month will not exceed 50% of the amount due by Customer for the applicable Covered Service for the applicable month. Financial Credits will be made in the form of a monetary credit applied to future use of the Service and will be applied within 60 days after the Financial Credit was requested.

SLA Exclusions

The SLA does not apply to any (a) features or Services designated pre-general availability (unless otherwise set forth in the associated Documentation); (b) features or Services excluded from the SLA (in the associated Documentation); or (c) errors (i) caused by factors outside of Google's reasonable control; (ii) that resulted from Customer's software or hardware or third party software or hardware, or both; (iii) that resulted from abuses or other behaviors that violate the Agreement; or (iv) that resulted from quotas listed in the Admin Console.

Google Kubernetes Engine Service Level Agreement (SLA)

During the Term of the agreement under which Google has agreed to provide Google Cloud Platform to Customer (as applicable, the "Agreement"), the Covered Service will provide a Monthly Uptime Percentage to Customer as follows (the "Service Level Objective" or "SLO"):

Covered Service	Monthly Uptime Percentage
Zonal Cluster (control plane)	99.5%
Regional Cluster (control plane)	99.95%
Autopilot Cluster (control plane)	99.95%
Autopilot Pods in Multiple Zones	99.9%
GKE Enterprise Autopilot Pods in Multiple Regions	99.99%

If Google does not meet the SLO, and if Customer meets its obligations under this SLA, Customer will be eligible to receive the Financial Credits described below. Monthly Uptime Percentage and Financial Credit are determined on a calendar month basis per cluster. This SLA states Customer's sole and exclusive remedy for any failure by Google to meet the SLO. Capitalized terms used in this SLA, but not defined in this SLA, have the meaning set forth in the Agreement. If the Agreement authorizes the resale or supply of Google Cloud Platform under a Google Cloud partner or reseller program, then all references to Customer in this SLA mean Partner or Reseller (as applicable), and any Financial Credit(s) will only apply for impacted Partner or Reseller order(s) under the Agreement.

Definitions

The following definitions apply to the SLA:

- **"Autopilot Pods in Multiple Zones"** means the compute capacity provisioned by the Google Kubernetes Engine Autopilot Service to schedule user pods, where pods are scheduled across two or more Zones in the same Region.

- **"GKE Enterprise Autopilot Pods in Multiple Regions"** means the compute capacity provisioned by the Google Kubernetes Engine Autopilot Service to schedule user pods, where pods are scheduled across two or more Regions in GKE Enterprise enabled clusters.

- **"Covered Service"** means:

  - Autopilot Pods in Multiple Zones; or

  - For each of Zonal Cluster (control plane), Regional Cluster (control plane), and Autopilot Cluster (control plane), the Kubernetes API provided by Customer's cluster(s), so long as the minor version of Google Kubernetes

Engine deployed in the cluster is a minor version currently offered in the Stable or Regular Channels.

- **“Downtime”** means:
  - For Autopilot Pods in Multiple Zones and GKE Enterprise Autopilot Pods in Multiple Regions, loss of Autopilot Cluster (control plane) connectivity to all applicable running pod instances. This Downtime does not include issues related to Load Balancing and VPN tunneling, which are covered under the respective Compute Engine and Cloud VPN SLAs.
  - For Zonal Cluster (control plane), Regional Cluster (control plane), and Autopilot Cluster (control plane), loss of external connectivity or Kubernetes API access to all applicable running clusters with the inability to launch replacement clusters in any Zone. This Downtime does not include:
    - Scheduled Downtime;
    - Loss of connectivity or other issues related to the underlying Compute Engine instances, such as Load Balancing and VPN tunneling, which are covered under the respective Compute Engine and Cloud VPN SLAs; or
    - Failure of Kubernetes nodes or the Kubernetes pods running on those nodes.
- **“Downtime Period”** means a period of five or more consecutive minutes of Downtime. Intermittent Downtime for a period of less than five minutes will not be counted towards any Downtime Periods.
- **“Financial Credit”** means the following:

Monthly Uptime Percentage			**Percentage of monthly bill for the respective Covered Service that does not meet SLO and that will be credited to Customer's future monthly bills**
Regional Cluster (control plane) or Autopilot Cluster (control plane)	**Zonal Cluster (control plane)**	**Autopilot Pods in Multiple Zones**	
99.0% to < 99.95%	99.0% to < 99.50%	99.0% to < 99.9%	10%

95.0% to < 99.0%	95.0% to < 99.0%	95.0% to < 99.0%	25%
< 95.0%	< 95.0%	< 95.0%	50%

- **"Monthly Uptime Percentage"** means total number of minutes in a month, minus the number of Downtime minutes suffered from all Downtime Periods in a month, divided by the total number of minutes in a month.

- **"Maintenance Window"** means a period of time when clusters are taken offline for maintenance tasks. This includes upgrading the Kubernetes APIs.

- **"Region"** means the applicable region described at https://cloud.google.com/compute/docs/regions-zones, as may be updated by Google from time to time.

- **"Regional Cluster (control plane)"** means a non-Autopilot cluster topology that consists of multiple replicas of the control plane, running in multiple Zones within a given Region, as described at https://cloud.google.com/kubernetes-engine/docs/concepts/regional-clusters.

- **"Regular Channel"** means the Regular release channel described at https://cloud.google.com/kubernetes-engine/docs/concepts/release-channels.

- **"Scheduled Downtime"** means Downtime resulting from Google performing maintenance on the Covered Service during a Maintenance Window.

- **"Stable Channel"** means the Stable release channel described at https://cloud.google.com/kubernetes-engine/docs/concepts/release-channels.

- **"Zonal Cluster (control plane)"** means a non-Autopilot single-Zone cluster with a single control plane (master) running in one Zone as described at https://cloud.google.com/kubernetes-engine/docs/how-to/creating-a-zonal-cluster.

- **"Zone"** means the applicable zone described at https://cloud.google.com/compute/docs/regions-zones/, as may be updated by Google from time to time.

Customer Must Request Financial Credit

In order to receive any of the Financial Credits described above, Customer must notify Google technical support within 30 days from the time Customer becomes eligible to receive a Financial Credit. Customer must also provide Google with server log files showing loss of external connectivity errors and the date and time those errors occurred. If Customer does not comply with these requirements, Customer will forfeit its right to receive a Financial Credit. If a dispute arises with respect to this SLA, Google will make a determination in good faith based on its system logs, monitoring reports, configuration records, and other available information.

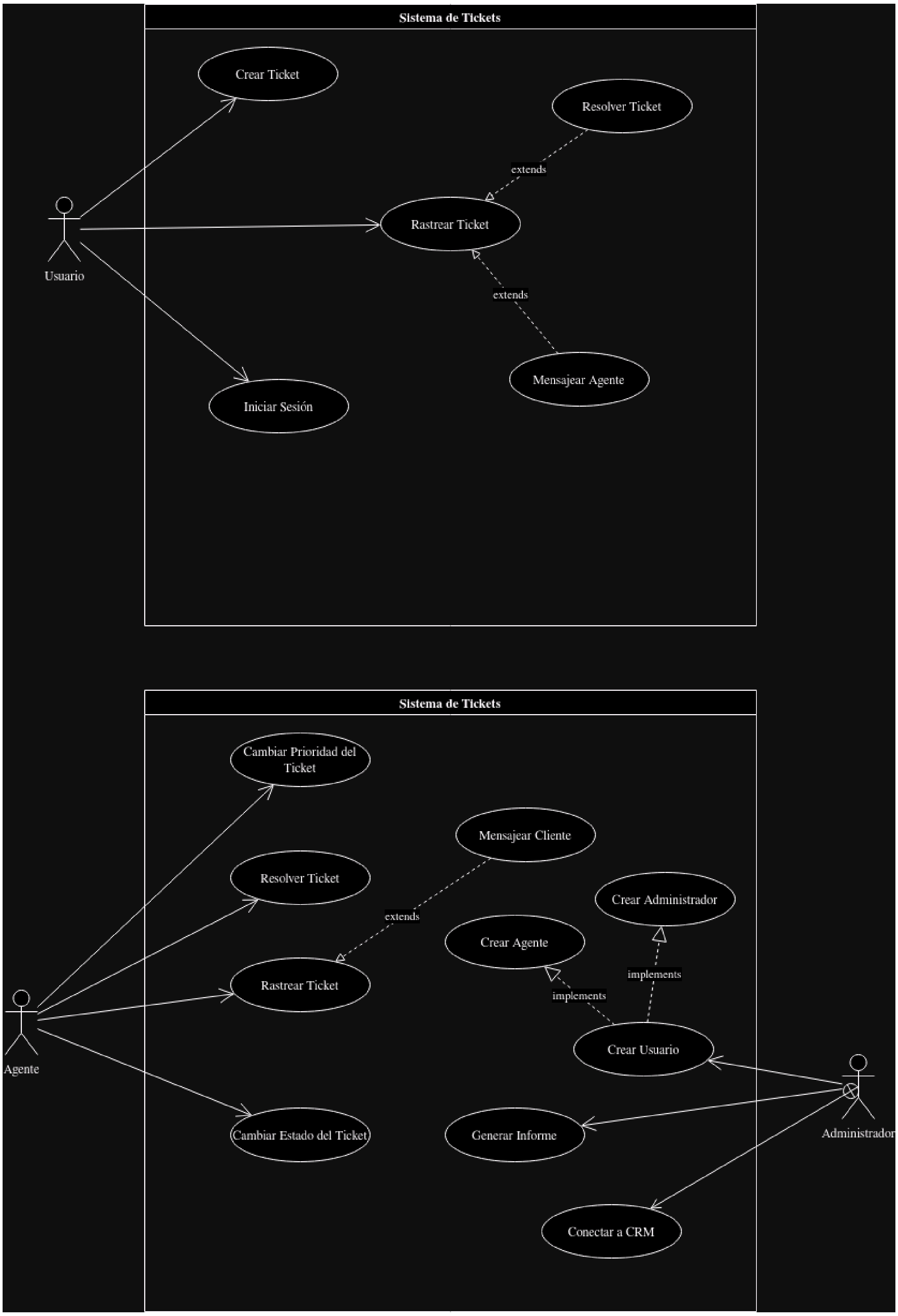
Maximum Financial Credit

The maximum aggregate number of Financial Credits issued by Google to Customer for all Downtime Periods in a single billing month will not exceed 50% of the amount due from Customer for the Covered Service for the applicable month. Financial Credits will be in the form of a monetary credit applied to future use of the Service and will be applied within 60 days after the Financial Credit was requested. Customer may not collect Financial Credits for the same episode of Downtime under both this SLA and another Google Cloud SLA.

SLA Exclusions

The SLA does not apply to any (a) features designated pre-general availability (unless otherwise set forth in the associated Documentation); (b) features excluded from the SLA (in the associated Documentation); (c) Customer clusters where the deployed minor version of Google Kubernetes Engine is not offered through the Stable or Regular Channels; or (d) errors: (i) caused by factors outside of Google's reasonable control; (ii) that resulted from Customer's software or hardware or third party software or hardware, or both; (iii) that resulted from abuses or other behaviors that violate the Agreement; or (iv) that resulted from quotas applied by the system or listed in the Admin Console.

CASOS DE USO


Sistema de Tickets
Crear Ticket
Resolver Ticket
extends
Usuario
Rastrear Ticket
extends
Mensajear Agente
Iniciar Sesión
Sistema de Tickets
Cambiar Prioridad del Ticket
Mensajear Cliente
Resolver Ticket
Crear Administrador
extends
Crear Agente
implements
Rastrear Ticket
implements
Agente
Crear Usuario
Administrador
Cambiar Estado del Ticket
Generar Informe
Conectar a CRM

DISEÑO DE BASE DE DATOS

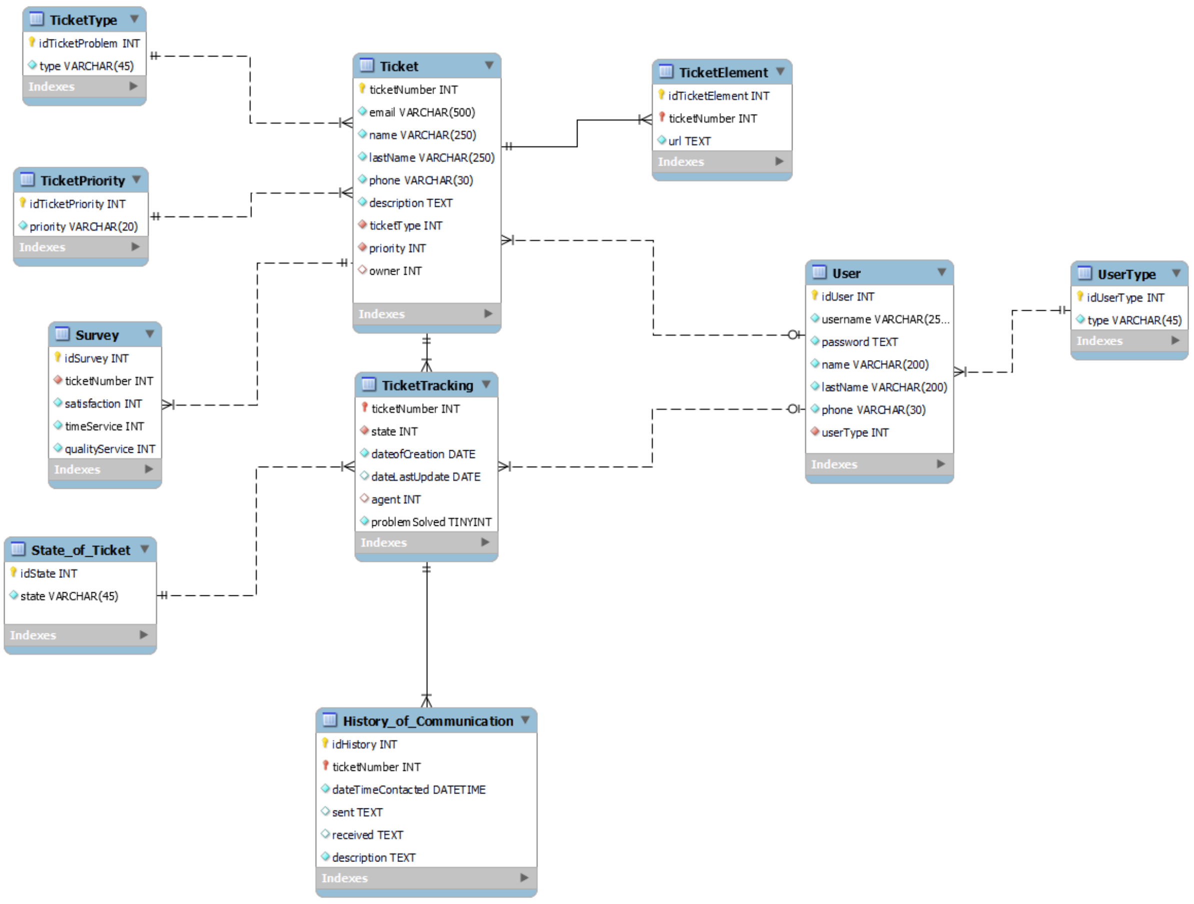
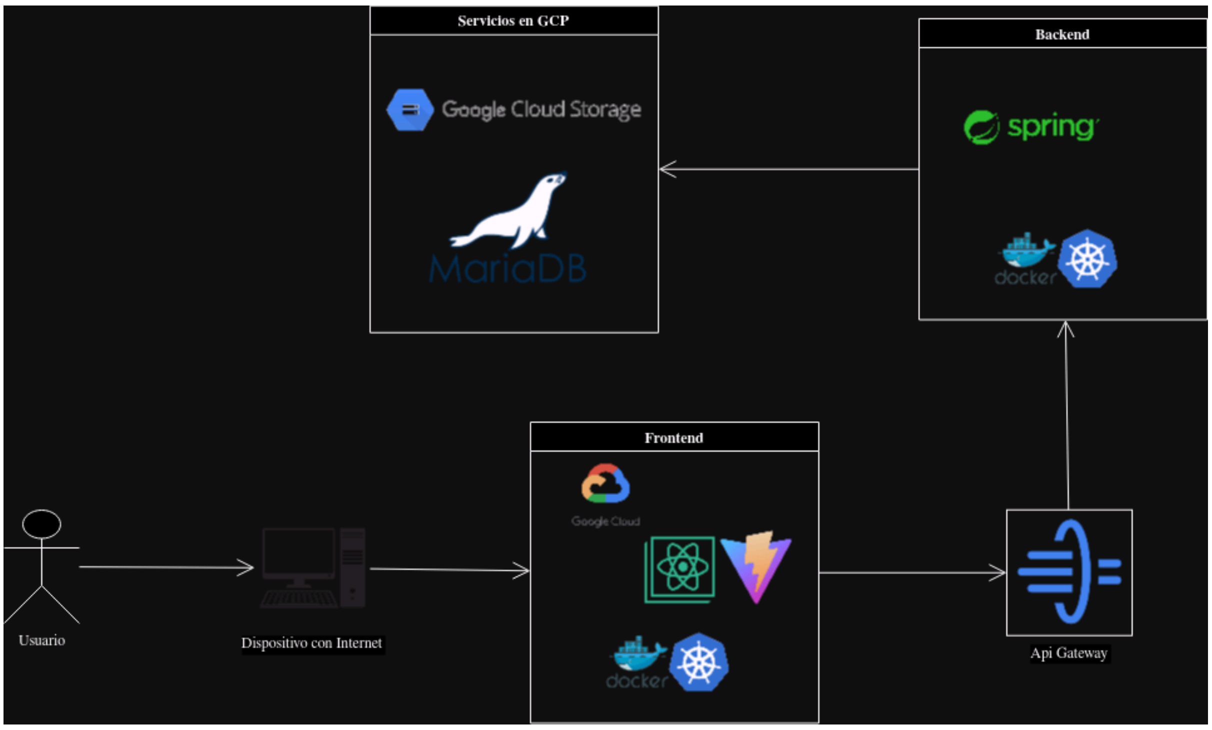


DIAGRAMA DE ARQUITECTURA


Servicios en GCP
Google Cloud Storage
MariaDB
Backend
spring
docker
Frontend
Google Cloud
docker
Usuario
Dispositivo con Internet
Api Gateway