UNIVERZITA KOMENSKÉHO V BRATISLAVE

SOFTWARE REQUIREMENTS SPECIFICATION

Alberto Cortés Herranz

Eric Ayestaran Guillorme

Renis Çenga

Xie Ailin

Subject: Development of Information Systems

Project Title: Backup System for Web Applications

Date: 05/11/2024

INDEX

1. Introduction

1.1 Purpose of Requirements Document

This document explains the functional and non-functional requirements for creating a Backup System for Web Applications. It serves as a binding agreement between the development team and the customer, ensuring that the system is developed in accordance with the client's needs and expectations. This document is intended for developers and administrators working on the project.

1.2 Scope of the Product

The backup system will automate and manage the process of backing up web applications running on different servers. It will include full and incremental backups, provide easy restoration in case of failure, and manage storage efficiently. The system is designed to work primarily in a Linux environment.

1.3 Definitions, Acronyms, and Abbreviations

- **Full Backup**: A complete copy of all data.
- **Incremental Backup**: A backup that only saves new or changed data since the last backup.
- **Administrator**: A user responsible for configuring and managing the backup system.
- **End User**: Users who rely on the web applications but do not directly interact with the backup system.
- **SFTP**: Secure File Transfer Protocol.

1.4 References

- IEEE/ISO/IEC 29148 Standard
- Client will provide documentation and data files.

1.5 Overview of the Remainder of the Document

The rest of this document provides a general description of the product, functional and non-functional requirements, specific use cases, and other relevant information pertaining to the backup system.

2. General Description

2.1 Product Perspective

This system is designed to manage backups for standalone web applications. It automates the backup process and allows flexible scheduling to ensure important data is always saved and easily restored if something goes wrong. The system works well with the client’s current setup, providing backup and recovery features without slowing down the web applications.

2.2 Product Functions

The Backup System will be able to back up and restore data to keep it safe and accessible. It can perform full backups of all data and databases, capturing everything about the web applications at times set by the administrator. It also supports incremental backups, saving only new or changed data since the last backup to save storage space.

Administrators can schedule backups based on each web application's needs, allowing flexible timing for full and incremental backups. They can perform selective backups by specifying files or folders to exclude, similar to how a (.gitignore) file works in Git.

The system supports both automatic and manual backups. Administrators can set up pre-backup and post-backup scripts that run before and after the backup process.

Backups are saved in common formats like zip, tar, or 7z, so other tools can read them. The system handles symbolic links correctly by storing them in the archive instead of following them during backup.

The system can store backup files on remote servers using SFTP. Users can enter remote server details like domain name or IP address, username, and target folder. Administrators can view backup statuses, including when each server was last backed up fully or incrementally, and see all stored backups for a server.

The system lets you set up servers for backup but keep them "disabled" until the administrator wants to start backups. It lets the administrator specifies in a settings file how many full backups to keep before old ones are automatically deleted, helping manage storage space.

2.3 User Characteristics

- **Administrator**: Manages the backup process, configures schedules, and restores data as needed.
- **End User**: Does not interact with the backup system but relies on its functionality to ensure the availability and integrity of web application data.

2.4 General Constraints

- The system must keep backups secure.
- Backups should not slow down the web applications.
- Only important data should be backed up to save storage space.
- The system is expected to work in a Linux environment

2.5 Assumptions and Dependencies

- The system will run on the existing server.
- Administrators have the necessary permissions and access to configure the backup system.
- Remote servers for storing backups are accessible via SFTP.

3. Specific Requirements

3.1 Functional Requirements

FR1. Full Backup
The system will be able to perform a complete backup of all data and databases periodically (e.g., weekly or monthly) as configured by the administrator.

FR2. Incremental Backup
The system will be able to perform incremental backups that save only the new or changed data since the last backup, optimizing storage space.

FR3. Backup Scheduling
The system must allow scheduling of full and incremental backups based on the needs of individual web applications as configured by the administrator.

FR4. Selective Backup
The system will allow administrators to specify a list of files or directories to exclude from backups, similar to the **.gitignore** file in Git, ensuring that only essential data is backed up.

FR5. Automated Backups
Backups must occur automatically based on the configured schedule set by the administrator, while also providing the option for manual initiation of backups when desired.

FR6. Data Restoration
The system should allow for easy restoration of backed-up data to the last saved state in the event of a failure.

FR7. Pre-Backup and Post-Backup Scripts
The system will provide the ability to specify pre-backup and post-backup scripts, which will be automatically executed before the backup starts and after it completes, respectively.

FR8. Backup Storage Format
Backups will be saved in formats that can be read by other tools, such as zip, tar, or 7z.

FR9. Handling of Symbolic Links
The system should be able to cope with symbolic links by storing or recovering them in/from the archive instead of following them while backing up.

FR10. Remote Backup Storage
The system will be able to store backup files on a remote server using protocols such as SFTP. Users will be able to specify the domain name or IP address of the server, username, and target path (folder).

FR11. Backup Status Display
The system will provide administrators with the ability to display the status of backups for all servers, including the last time each was backed up fully and incrementally, as well as a list of all currently stored backups for a particular server.

FR12. Manual Backup
Administrators will have the option to perform backups manually, in addition to the automated scheduled backups.

FR13. Configurable Backup Targets

Administrators can set up servers for backup but keep them "disabled." This means the backup settings are saved, but the backups won't actually run until the administrator decides to turn them on.

FR14. Configuring How Many Backups to Keep

The system will let administrators specify in a settings file how many full backups to keep. Once this number is reached, old backups will be automatically deleted. This helps manage storage space effectively.

FR15. Operating System Compatibility
The system is expected to work in a Linux environment.

3.2 Non-Functional Requirements

NFR1. Storage Efficiency
The system must manage storage space effectively by retaining only necessary backups. It should allow administrators to specify in a configuration file how many full backups are kept before they are automatically deleted. Old incremental backups should be deleted after a full backup is completed to optimize storage usage.

NFR2. Performance
Backup operations must not interfere with the normal operation of web applications, ensuring minimal impact on system resources.

NFR3. Reliability
The system must consistently perform backups without failure and ensure accurate data restoration.

NFR4. Usability
The backup system should have a user-friendly interface that allows administrators to easily configure schedules, manage backup settings, and restore data.

NFR6. Scalability
The system should be able to support backups for multiple web applications and servers without performance degradation.

NFR7. Compatibility
The system is expected to work in a Linux environment.

UNIVERZITA KOMENSKÉHO V BRATISLAVE

BACKUP SYSTEM – SYSTEM DESING

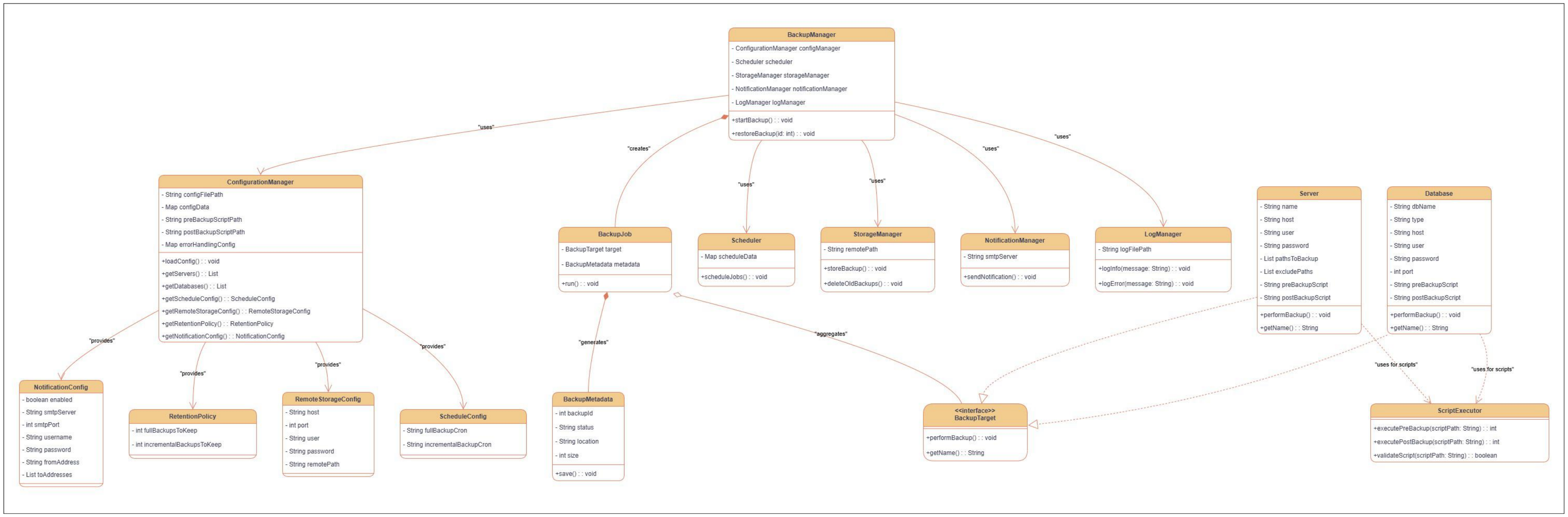
Renis Çenga

Alberto Cortés Herranz

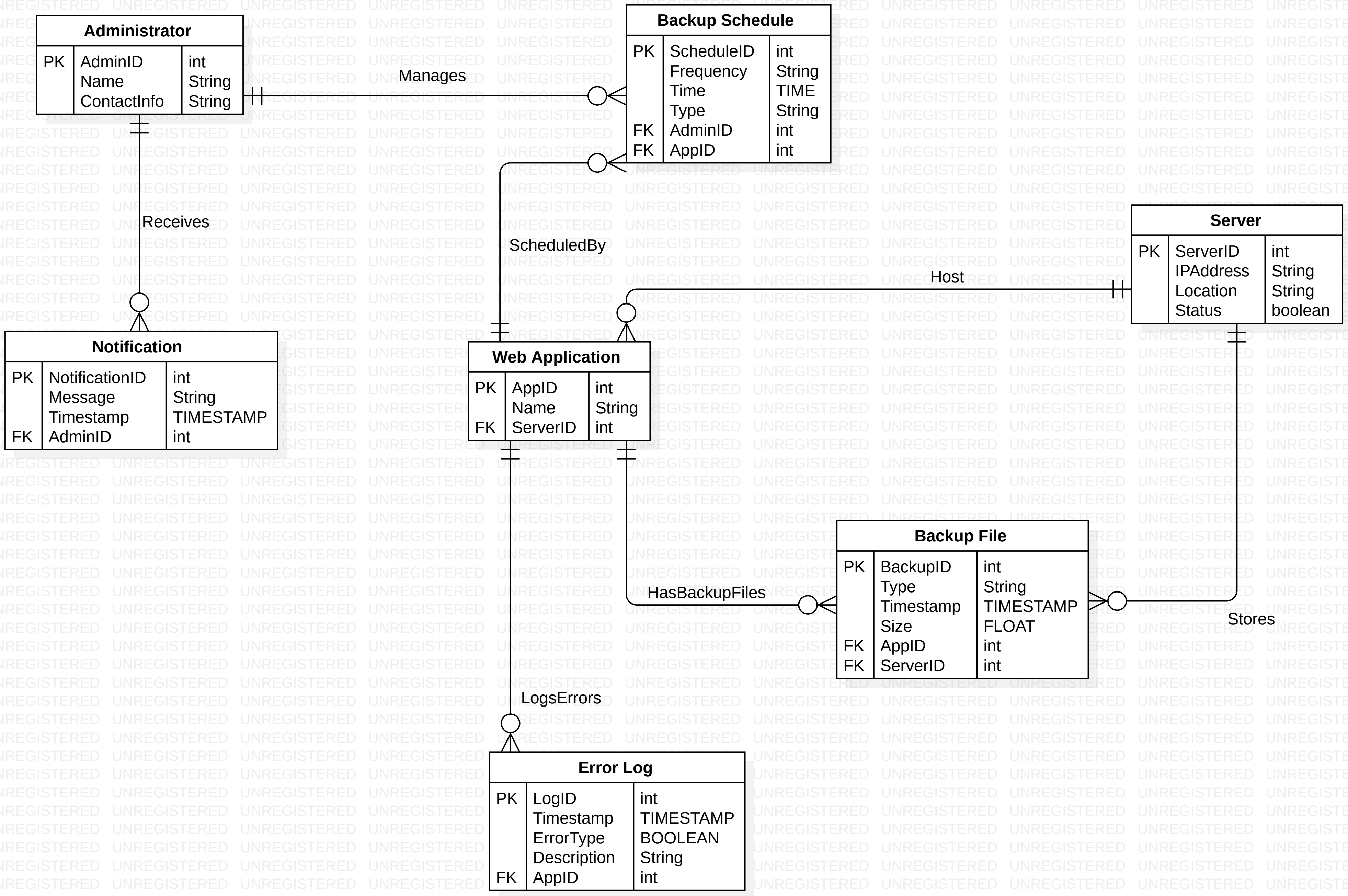
Eric Ayestaran Guillorme

Subject: Development of Information Systems
Project Title: Backup System for Web Applications

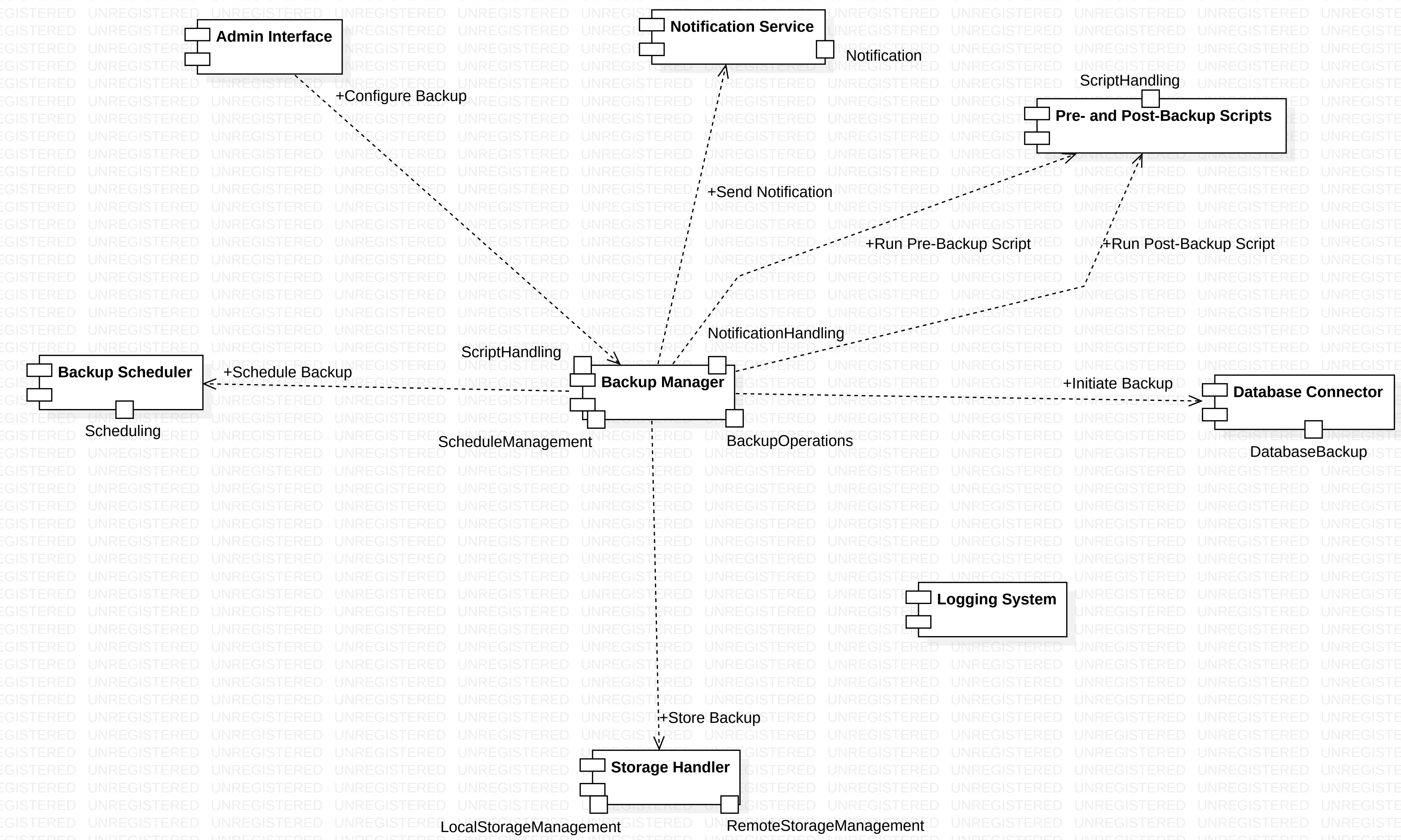
UML Class Diagram



ERD Diagram


Administrator
PK AdminID int
Name String
ContactInfo String
Manages
Backup Schedule
PK ScheduleID int
Frequency String
Time TIME
Type String
FK AdminID int
FK AppID int
Receives
ScheduledBy
Server
PK ServerID int
IPAddress String
Location String
Status boolean
Host
Notification
PK NotificationID int
Message String
Timestamp TIMESTAMP
FK AdminID int
Web Application
PK AppID int
Name String
FK ServerID int
Backup File
PK BackupID int
Type String
Timestamp TIMESTAMP
Size FLOAT
FK AppID int
FK ServerID int
HasBackupFiles
Stores
LogsErrors
Error Log
PK LogID int
Timestamp TIMESTAMP
ErrorType BOOLEAN
Description String
FK AppID int

Component Diagram



User Interface Design

Introduction

This document explains the details of the Backup System. It includes how the command-line interface (CLI) works, the available commands, how scripts are used. The design follows the structure of our system diagrams, including UML, component, and data models.

Command-Line Interface (CLI) Specification

The CLI allows administrators to manage the Backup System using commands. It supports command-line arguments for flexibility and provides features like scheduling backups, running them manually, restoring data, and configuring settings.

Available Commands

Command	*Description*	*Syntax*
help	Displays the list of available commands.	backup-system help
schedule_backup	Configures a new backup schedule.	backup-system schedule_backup --target <TARGET_NAME> --type <TYPE> --schedule "<CRON_EXPRESSION>"
run_backup	Manually triggers a backup for a specific target.	backup-system run_backup --target <TARGET_NAME>
restore_backup	Restores data from a backup.	backup-system restore_backup --id <BACKUP_ID>
status	Displays the current status of the backup system.	backup-system status
list_backups	Lists all stored backups with metadata.	backup-system list_backups
enable_server	Enables a server for backups.	backup-system enable_server --target <TARGET_NAME>
disable_server	Disables a server, stopping backups for it.	backup-system disable_server --target <TARGET_NAME>
validate_script	Validates pre- and post-backup scripts to ensure they are functional.	backup-system validate_script --path <SCRIPT_PATH>
exit	Exits the program.	exit

Detailed Command Syntax and Examples

1. schedule_backup

- **Purpose**: Configure a new schedule for backups.
- **Syntax**:
- backup-system schedule_backup --target <TARGET_NAME> --type <TYPE> --schedule "<CRON_EXPRESSION>"
- **Arguments**:
  - --target: Name of the backup target (e.g., Server1, Database1).
  - --type: Type of backup (full or incremental).
  - --schedule: Cron-like schedule (e.g., "0 2 * * 0" for every Sunday at 2 AM).
- **Example**:
- backup-system schedule_backup --target Server1 --type full --schedule "0 3 * * 1"
  - This schedules a **full backup** for Server1 every Monday at 3 AM.

2. run_backup

- **Purpose**: Manually triggers a backup for a specific target.
- **Syntax**:
- backup-system run_backup --target <TARGET_NAME>
- **Arguments**:
  - --target: Name of the target to back up (e.g., Server1 or Database1).
- **Example**:
- backup-system run_backup --target Database1
  - This runs a backup for Database1 immediately.

3. restore_backup

- **Purpose**: Restore data from a specific backup.
- **Syntax**:
- backup-system restore_backup --id <BACKUP_ID>
- **Arguments**:
  - --id: ID of the backup to restore.

- **Example**:
- backup-system restore_backup --id 12345
    - This restores the backup with ID 12345.

5. validate_script

- **Purpose**: Ensures the provided script works before assigning it.
- **Syntax**:
- backup-system validate_script --path <SCRIPT_PATH>
- **Arguments**:
    - --path: Path to the script to validate.
- **Example**:
- backup-system validate_script --path /scripts/pre_backup.sh
- **Output**:
    - Success: "Script validated successfully."
    - Failure: "Script validation failed: Syntax error at line 10."

Integration of Scripts

Purpose

Scripts allow administrators to perform specific tasks before and after backups, such as stopping services or cleaning up temporary files.

Configuration

Scripts are defined in the configuration file, like this:

```
servers:
  - name: Server1
    pre_backup_script: /scripts/pre_backup_server1.sh
    post_backup_script: /scripts/post_backup_server1.sh
databases:
  - name: Database1
    pre_backup_script: /scripts/pre_backup_db1.sh
    post_backup_script: /scripts/post_backup_db1.sh
```

UNIVERZITA KOMENSKÉHO V BRATISLAVE

BACKUP SYSTEM – TEST SCENARIOS

Renis Çenga

Alberto Cortés Herranz

Eric Ayestaran Guillorme

Subject: Development of Information Systems
Project Title: Backup System for Web Applications

1. SCHEDULING AND RUNNING A FULL BACKUP

Scenario A: Scheduling a Full Backup for Server1

1. **Action**: The user opens the CLI and enters:

```
backup-system schedule_backup --target Server1 --type full --schedule "0 3 * * 0"
```

Result: The system sets up a cron job for a full backup of “Server1” every Sunday at 3 AM.

2. **Action**: The user types status in the CLI.
   Result: The CLI shows that a scheduled job for Server1 will run each Sunday at 3 AM.

Scenario B: Manually Running a Full Backup

1. **Action**: The user enters:

```
backup-system run_backup --target Server1
```

Result: The system immediately starts a **full** backup for Server1 (assuming full is the default if no --type is specified).

2. **Action**: The user watches the CLI output.
   Result: It displays logs about compressing files and, once finished, shows “Backup completed successfully.”

2. SCHEDULING AND RUNNING AN INCREMENTAL BACKUP

Scenario A: Scheduling an Incremental Backup for Database1

1. **Action**: The user opens the CLI and enters:

```
backup-system schedule_backup --target Database1 --type incremental --schedule "0 1 * * *"
```

Result: The system schedules an incremental backup of Database1 every day at 1 AM.

2. **Action**: The user types list_backups.
   Result: No previous backups appear for Database1 yet, since it hasn’t run.

Scenario B: Manually Running an Incremental Backup

1. **Action**: The user enters:

```
backup-system run_backup --target Database1 --type incremental
```

Result: The system performs an incremental dump for Database1, compresses it, and logs success.

2. **Action**: The user runs list_backups again.
   Result: The new incremental backup shows up with its backup ID, status, and file location.

3. RESTORING FROM A PREVIOUS BACKUP

Scenario: Listing and Choosing a Backup to Restore

1. **Action**: The user enters list_backups in the CLI.
   Result: The system lists all known backups, showing **backupId**, **targetName**, **backupType**, and **timestamp**.
2. **Action**: The user picks a backup ID (e.g., 12345).
   Result: The system is ready to restore from this ID.
3. **Action**: The user enters:

```
backup-system restore_backup --id 12345
```

Result: The system finds the archive with backupId=12345, restores it to the original location, and logs "Restore completed successfully."

4. ENABLING AND DISABLING A SERVER

Scenario A: Disabling a Server

1. **Action**: The user types:

```
backup-system disable_server --target Server1
```

Result: The system marks Server1 as disabled so it won't be backed up.

2. **Action**: The user tries to run a backup on Server1:

```
backup-system run_backup --target Server1
```

Result: The CLI says "Server1 is disabled. Backup aborted."

Scenario B: Enabling the Server Again

1. **Action**: The user types:

```
backup-system enable_server --target Server1
```

Result: The system sets Server1 to enabled.

2. **Action**: The user runs a backup on Server1 again:

```
backup-system run_backup --target Server1
```

Result: This time, the backup proceeds successfully.

5. VALIDATING PRE/POST-BACKUP SCRIPTS

Scenario: Validate a Script

1. **Action**: The user enters:

```
backup-system validate_script --path /scripts/pre_backup_server1.sh
```

Result: The system tests the script. If it returns exit code 0, the CLI shows "Script validated successfully."

6. CHECKING SYSTEM STATUS

Scenario: Viewing All Scheduled Jobs

1. **Action**: The user enters status in the CLI.
   Result: The system shows:
   - Whether the scheduler is active.
   - Next run times for each scheduled backup.
   - Last completed backup times.
2. **Action**: The user verifies the displayed information matches recent backups and cron schedules.
   Result: Everything matches correctly.

7. BACKUP WITH SYMBOLIC LINKS

Scenario: Handling Symbolic Links in a Backup

1. **Action**: The user creates a directory with symbolic links and sets it as a backup target (e.g., Server2).
2. **Action**: The user enters:

```
backup-system run_backup --target Server2
```

Result: The system correctly stores the **links** themselves in the archive rather than copying the original files multiple times. The CLI logs “Backup completed successfully.”

8. BACKUP UPLOAD TO REMOTE SFTP STORAGE

Scenario: Uploading a Backup to an SFTP Server

1. **Action**: The user configures valid SFTP credentials in the config.yaml (host, user, password, etc.).
2. **Action**: The user runs any backup (e.g., run_backup --target Server1).
3. **Action**: The system finishes compressing the backup and attempts to upload it.
   Result: The CLI logs “Backup uploaded to remote location” if everything is correct. If credentials fail, it logs an error.

9. INVALID CONFIGURATION ERROR HANDLING

Scenario: Starting the System with a Bad Config

1. **Action**: The user supplies a configuration file missing required fields (like remoteStorage.host or a cron expression).
2. **Action**: The user tries to run the system
   Result: The system detects that required fields are missing, logs an error (e.g., “Error: Missing host for remote storage”), and does **not** proceed with backups or scheduling.