



# Development of a Change Data Capture (CDC) tool Medical care register

**Name:** Muhammad Hassan

**Matrikelnummer:** 1503186

**Role:** Backend developer & DevOps

**Course:** Software Development Project (AI5094)

**Presented to:** Prof. Michael Jahn



This project involved the development of a functioning Clinical Data Collection (CDC) application focused on gathering patient data for diabetic foot syndrome in the form of a digital register.

The goal was to design and implement a software solution that supports structured, secure, and efficient data entry, management, and retrieval to assist healthcare professionals in monitoring and research.

The project was developed collaboratively, with each team member contributing to a specific functional area (backend, frontend, DevOps, etc.), and documented through regular reports and presentations.



- **Framework:** Express.js (Runtime engine : Node.js)
- **ORM:** Knex.js with Objection.js
- **Database:** MariaDB
- **Validation:** Joi schema validators
- **Error Handling:** Custom success/error response classes
- **Logging:** Pino Asynchronous Logger

# My Role in the Project



- Implementation of Questionnaire module
- Gitlab Repository management
- Backend boiler code setup
- Backend DevOps

# Agile project management



Search or go to...

Project

Pinned

Issues 7

Merge requests 0

Manage

Plan

Issues 7

Issue boards

Milestones

Wiki

Code

Build

Secure

Help

Project group 06 / Info / Issues

Open 0 Closed 5 All 5

Assignee is Muhammad Hassan

Created date

Questionnaire module

#19 · created 2 days ago by Muhammad Hassan

Closed

1

closed 1 minute ago

Patient Module

#18 · created 1 month ago by Muhammad Hassan

Closed

closed 1 minute ago

Setup DevOps

#12 · created 1 month ago by Muhammad Hassan

Closed

2

closed 1 month ago

Webserver Configuration

#8 · created 1 month ago by Huzaifa Khatri

Closed

1

closed 1 month ago

medical database

#1 · created 2 months ago by Michael Jahn

Closed

1

closed 1 month ago

Show 100 items



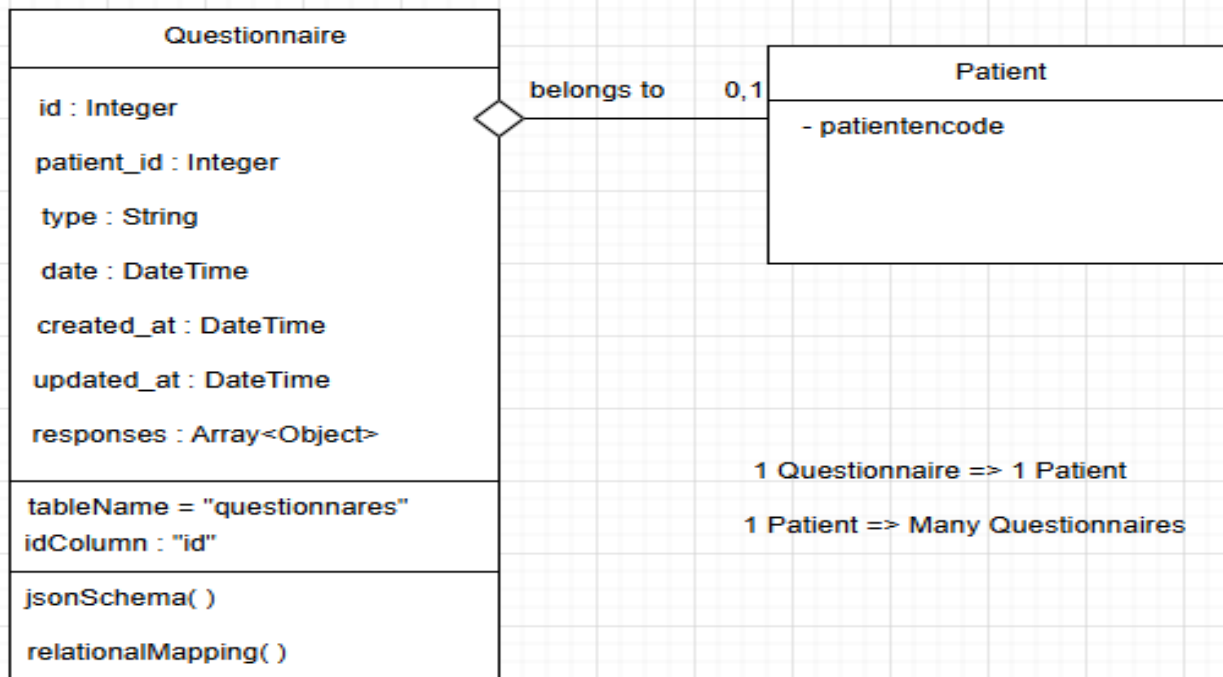
**Purpose of the module:** This module handles the creation, retrieval, and updating of patient questionnaires related to diabetic foot syndrome. It ensures secure, validated, and structured data entry in the CDC register.

## Key Features:

- **Create Questionnaire**
  - Ensures "pre" type is unique per patient
  - Validates schema before insertion
  - Adds records with timestamps



- **Retrieve Questionnaire by patient ID**
  - Returns all questionnaires linked to a specific patient
- **Update Questionnaire**
  - Validates ownership
  - Update 'responses' and 'updated\_at'



Class Diagram (Questionnaire)



# Source code



```
14
15 export const addQuestionnaire = async (req, res, next) => {
16   const { patient_id, type, responses, date } = req.body;
17
18   try {
19     await addQuestionnaireValidator.validateAsync(req.body).catch((err) => {
20       throw CustomError.validation(err.message);
21     });
22
23     const patient = await Patient.query().findOne({
24       patientcode: patient_id,
25     });
26
27     if (!patient) {
28       throw CustomError.notFound("Patient not found with this id");
29     }
30
31     if (type === "pre") {
32       const existingPreRecord = await Questionnaire.query().findOne({
33         patient_id,
34         type: "pre",
35       });
36
37       if (existingPreRecord) {
38         throw CustomError.badRequest(
39           "A 'pre' questionnaire already exists for this patient"
40         );
41       }
42     }
43
44     const insertedRecord = await Questionnaire.query().insert({
45       patient_id,
46       type,
47       responses,
48       date,
49     });
50
51     return next(
52       CustomSuccess.createSuccess(
53         insertedRecord,
54         "Questionnaire added successfully",
55         200
56       )
57     );
58   } catch (err) {
59     logger.error(err.message);
60     next(err);
61   }
62 }
```

```
export const getQuestionnaireByPatientId = async (req, res, next) => {
  const { patient_id } = req.params;

  try {
    await getQuestionnaireByPatientValidator
      .validateAsync(req.params)
      .catch((err) => {
        throw CustomError.validation(err.message);
      });

    const questionnaire = await Questionnaire.query().where({ patient_id });

    if (questionnaire.length == 0) {
      throw CustomError.notFound(
        "No questionnaire found for patient with ID ${patient_id}"
      );
    }

    return next(
      CustomSuccess.createSuccess(
        questionnaire,
        "Questionnaire retrieved successfully",
        200
      )
    );
  } catch (err) {
    logger.error(err.message);
    next(err);
  }
};

export const updateQuestionnaire = async (req, res, next) => {
  const { questionnaire_id } = req.params;
  const { patient_id, responses } = req.body;

  try {
    await updateQuestionnaireValidator
      .validateAsync({
        questionnaire_id,
        patient_id,
        responses,
      })
      .catch((err) => {
        throw CustomError.validation(err.message);
      });

    const questionnaire = await Questionnaire.query().findById(
      questionnaire_id
    );

    if (!questionnaire) {
      throw CustomError.notFound("Questionnaire not found.");
    }

    if (questionnaire.patient_id !== patient_id) {
      throw CustomError.validation(
        "Patient ID does not match the questionnaire's owner."
      );
    }
  }
};
```



- **CI/CD Pipeline Setup (GitLab):**

- Implemented `.gitlab-ci.yml` for defining pipeline stages
- Configured a self-hosted GitLab runner to execute the pipeline on the target server
- Set up Git remote URL with embedded **Personal Access Token** for secure repository access
- Set up secure SSH access:
  - Generated SSH keys and configured permissions
  - Added public key to `authorized_keys`
  - Stored private key in GitLab CI/CD variables

- **Deployment Automation:**

- Pipeline triggers post deployment script that pulls the latest commit and uses PM2 to start and manage the server



```
gitlab-ci.yml x JS Questionnaire.js JS QuestionnaireRouter.js JS QuestionnaireValidator.js JS QuestionnaireController.js
gitlab-ci.yml
You, last month | 1 author (You)
1 stages:
2   - deploy
3
4 deploy_dev:
5   stage: deploy
6   tags:
7     - ci-cd-job-runner
8   only:
9     - dev
10  before_script:
11    - echo "== BEFORE SCRIPT START =="
12    - which ssh-agent || ( apt-get update -y && apt-get install openssh-client -y )
13    - eval $(ssh-agent -s)
14    - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
15    - mkdir -p ~/.ssh
16
17  script:
18    - ssh -o StrictHostKeyChecking=no pg06-web@pg06.regifor.de 'bash /home/deploy_scripts/deploy_dev.sh'
19
```

## Pipeline configuration

```
MINGW64/c/Users/Hassan/Downloads/info-priv-key
GNU nano 7.2 pg06.regifor.de-le-ssl.conf
<IfModule mod_ssl.c>
<VirtualHost *:443>
  ServerAdmin administrator@regifor.de

  ServerName pg06.regifor.de
  ServerAlias www.pg06.regifor.de

  DocumentRoot /var/www/pg06.regifor.de

  ErrorLog ${APACHE_LOG_DIR}/pg06.regifor.de-error.log
  CustomLog ${APACHE_LOG_DIR}/pg06.regifor.de-access.log combined

  ProxyPass "/api/" "http://localhost:8000/api/"
  ProxyPassReverse "/api/" "http://localhost:8000/api/"

  ProxyPass "/" "http://localhost:3000/"
  ProxyPassReverse "/" "http://localhost:3000/"

  SSLCertificateFile /etc/letsencrypt/live/pg06.regifor.de/fullchain.pem
  SSLCertificateKeyFile /etc/letsencrypt/live/pg06.regifor.de/privkey.pem
  Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
</IfModule>

Help Write Out Where Is Cut Execute Location N=U Undo
Exit Read File Replace AU Paste AJ Justify AV Go To Line M=E Redo
```

## Setting up Reverse proxy using Apache2



```
MINGW64/c/Users/Hassan/Downloads/info-priv-key
GNU nano 7.2 /etc/sudoers
# Per-user preferences; root won't have sensible values for them.
#Defaults:sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults:sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
#Defaults:sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@include /etc/sudoers.d

# for deployment purpose
pg06-web ALL=(ALL) NOPASSWD: /usr/bin/git, /usr/bin/npm, /usr/local/bin/pm2
```

```
pg06-web@lnx-nbg1-03:~/.ssh$ cd ..
pg06-web@lnx-nbg1-03:~$ sudo pm2 ls
```

id	name	mode	u	status	cpu	memory
1	pg-06-backend	fork	4	online	0%	69.7mb
0	pg-06-frontend	fork	4	online	0%	67.6mb

```
pg06-web@lnx-nbg1-03:~$
```

Configure PM2, npm and git to work  
with escalated privileges

Instances running with PM2



```
MINGW64:/c:/Users/Hassan/Downloads/info-priv-key
GNU nano 7.2 deploy_dev.sh
#!/bin/bash

cd /var/www/pg06-server/info || exit 1

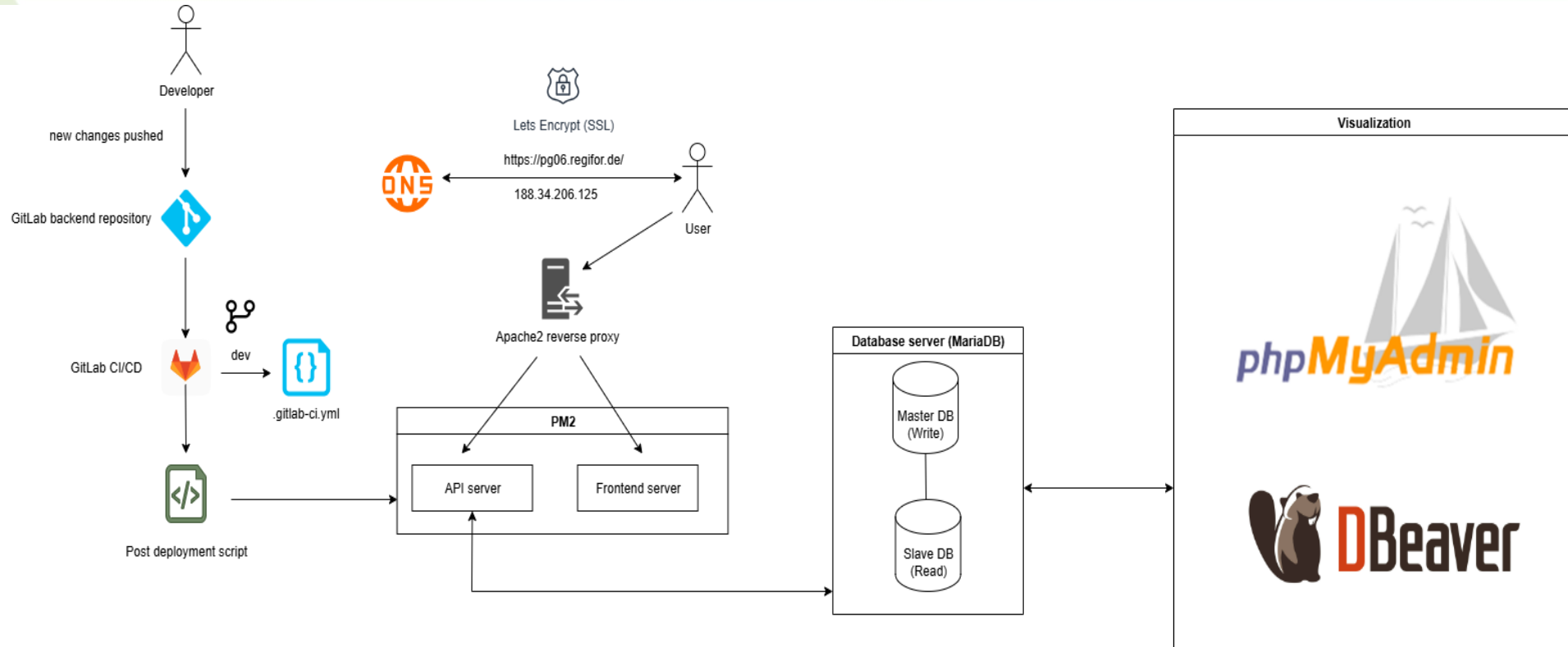
sudo git pull origin dev

sudo npm install

sudo pm2 restart pg-06-backend || sudo pm2 start "npm run dev" --name pg-06-backend

[ File 'deploy_dev.sh' is unwritable ]...
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File ^_ Replace   ^P Paste     ^J Justify   ^_ Go To Line M-E Redo
```

Post-deployment script



High Level Architecture Diagram



Thank you.  
Questions?