1. RDS

1.1. Bảng Candidate

**CREATE** **TABLE** public.candidates (

id **int4** **NOT** **NULL** **GENERATED** **BY** **DEFAULT** **AS** **IDENTITY**,

**age** **int4** **NOT** **NULL** **DEFAULT** 18,

home\_town **varchar** **NOT** **NULL**,

university **varchar** **NOT** **NULL**,

condition\_result **varchar** **NULL**,

**CONSTRAINT** candidates\_pk **PRIMARY** **KEY** (id)

);

1.2. Bảng Conditions

**CREATE** **TABLE** public.conditions (

id **int4** **NOT** **NULL** **GENERATED** **BY** **DEFAULT** **AS** **IDENTITY**,

operate **varchar** **NOT** **NULL**,

target\_rule **varchar** **NOT** **NULL**,

operate\_match **varchar** **NOT** **NULL**,

**input** **varchar** **NOT** **NULL**,

question\_id **integer** **NOT** **NULL**,

**CONSTRAINT** conditions\_pk **PRIMARY** **KEY** (id)

);

Điều kiện bổ sung:

* operate chỉ nhận các giá trị: IF, AND, OR
* target\_rule chỉ nhận các giá trị: age, location, university
* operate\_match chỉ nhận các giá trị: >, >=, <, <= , ==, !=

Các giá trị trên sẽ được định nghĩa trong enum của code

namespace FlowApp.Models

{

public abstract class BaseEnum

{

public string Code { get; set; }

public string Name { get; set; }

protected BaseEnum() { }

public BaseEnum(string code, string name)

{

Code = code;

Name = name;

}

public abstract bool IsValidCode(string code);

}

}

namespace FlowApp.Models

{

public class Match : BaseEnum

{

protected Match() { }

public Match(string code, string name) : base(code, name)

{

}

public static Match Is = new Match("==","Is");

public static Match IsGreaterThan = new Match(">", "Is greater than");

public static Match IsGreaterThanOrEqualTo = new Match(">=", "Is greater than or equal to");

public static Match IsLessThan = new Match("<", "Is less than");

public static Match IsLessThanOrEqualTo = new Match("<=", "Is less than or equal to");

public static Match IsNotEqualTo = new Match("!=", "Is not equal to");

public static List<Match> Matches = new List<Match>()

{

Is,

IsGreaterThan,

IsGreaterThanOrEqualTo,

IsLessThan,

IsLessThanOrEqualTo,

IsNotEqualTo

};

public override bool IsValidCode(string code)

{

var item = Matches.FirstOrDefault(x => x.Code == code);

return item is not null;

}

public static bool IsValid(string code)

{

return new Match().IsValidCode(code);

}

}

}

namespace FlowApp.Models

{

public class Operate : BaseEnum

{

protected Operate() { }

public Operate(string code, string name) : base(code, name)

{

}

public static Operate If = new Operate("IF","If");

public static Operate And = new Operate("AND","And");

public static Operate Or = new Operate("OR","Or");

public static List<Operate> Operates = new List<Operate>() { If, And, Or };

public override bool IsValidCode(string code)

{

var item = Operates.FirstOrDefault(x => x.Code == code);

return item is not null;

}

public static bool IsValid(string code)

{

return new Operate().IsValidCode(code);

}

}

}

namespace FlowApp.Models

{

public class Rule : BaseEnum

{

protected Rule() { }

public Rule(string code, string name) : base(code, name)

{

}

public static Rule Age = new Rule("AGE","Age");

public static Rule Location = new Rule("LOCATION", "Location");

public static Rule University = new Rule("UNIVERSITY", "University");

public static List<Rule> Rules = new List<Rule>() { Age, Location, University };

public override bool IsValidCode(string code)

{

var item = Rules.FirstOrDefault(x => x.Code == code);

return item is not null;

}

public static bool IsValid(string code)

{

return new Rule().IsValidCode(code);

}

}

}

namespace FlowApp.Models

{

public class Question

{

public int QuestionId { get; set; }

public string QuestionContent { get; set; }

public static List<Question> Items = new List<Question>()

{

new Question(){QuestionId = 1, QuestionContent = "Age?" },

new Question(){QuestionId = 2, QuestionContent = "Hometown?" },

new Question(){QuestionId = 3, QuestionContent = "Which university they graduated from?" },

};

}

}

2. Service

2.1. Phương thức làm việc với database

public interface IRepository

{

IQueryable<Candidate> Candidates { get; }

IQueryable<Condition> Conditions { get; }

void Add(Candidate candidate);

void Update(Candidate candidate);

void Delete(Candidate candidate);

void Add(Condition candidate);

void Update(Condition candidate);

void Delete(Condition candidate);

Task<bool> SaveChangesAsync();

}

public class Repository : IRepository

{

private readonly DataContext \_context;

public Repository(DataContext context)

{

\_context = context;

}

public IQueryable<Candidate> Candidates => \_context.Candidates;

public IQueryable<Condition> Conditions => \_context.Conditions;

public void Add(Candidate candidate)

{

\_context.Candidates.Add(candidate);

}

public void Add(Condition condition)

{

\_context.Conditions.Add(condition);

}

public void Delete(Candidate candidate)

{

\_context.Entry(candidate).State = EntityState.Deleted;

}

public void Delete(Condition condition)

{

\_context.Entry(condition).State = EntityState.Deleted;

}

public async Task<bool> SaveChangesAsync()

{

var result = await \_context.SaveChangesAsync();

return result > 0;

}

public void Update(Candidate candidate)

{

\_context.Entry(candidate).State = EntityState.Modified;

}

public void Update(Condition condition)

{

\_context.Entry(condition).State = EntityState.Modified;

}

}

2.1. Add Condition

[HttpPost]

public async Task<IActionResult> Post([FromBody] ConditionModel request)

{

// validation input

if (!Operate.IsValid(request.OperateCode))

{

ModelState.AddModelError("", "Invalid operate");

}

if (!Rule.IsValid(request.TargetRuleCode))

{

ModelState.AddModelError("", "Invalid rule");

}

if (!Match.IsValid(request.OperateMatchCode))

{

ModelState.AddModelError("", "Invalid match");

}

var question = Question.Items.FirstOrDefault(x => x.QuestionId == request.QuestionId);

if(question is null)

{

ModelState.AddModelError("", "Question not found");

}

if (ModelState.IsValid)

{

// khởi tạo đối tượng

var condition = new Condition()

{

Input = request.Input,

QuestionId = request.QuestionId,

Operate = request.OperateCode,

OperateMatch = request.OperateMatchCode,

TargetRule = request.TargetRuleCode

};

\_repository.Add(condition);

// lưu đối tượng vào database

var result = await \_repository.SaveChangesAsync();

if (result)

{

// trả về mã HTTP Status 204 thông báo lưu thành công

return NoContent();

}

}

// trả về mã lỗi 400 thông báo dữ liệu không hợp lệ

return BadRequest();

}

2.2. Edit Condition

[HttpPut]

public async Task<IActionResult> Put(int id, [FromBody] ConditionModel request)

{

// check condition exist

var condition = \_repository.Conditions.FirstOrDefault(x => x.Id == id);

if(condition is null)

{

return NotFound();

}

// validation input

if (!Operate.IsValid(request.OperateCode))

{

ModelState.AddModelError("", "Invalid operate");

}

if (!Rule.IsValid(request.TargetRuleCode))

{

ModelState.AddModelError("", "Invalid rule");

}

if (!Match.IsValid(request.OperateMatchCode))

{

ModelState.AddModelError("", "Invalid match");

}

var question = Question.Items.FirstOrDefault(x => x.QuestionId == request.QuestionId);

if (question is null)

{

ModelState.AddModelError("", "Question not found");

}

if (ModelState.IsValid)

{

condition.Input = request.Input;

condition.QuestionId = request.QuestionId;

condition.Operate = request.OperateCode;

condition.OperateMatch = request.OperateMatchCode;

condition.TargetRule = request.TargetRuleCode;

\_repository.Update(condition);

var result = await \_repository.SaveChangesAsync();

if (result)

{

return NoContent();

}

}

return BadRequest();

}

2.3. Delete Condition

[HttpDelete]

public async Task<IActionResult> Delete(int id)

{

// check condition exist

var condition = \_repository.Conditions.FirstOrDefault(x => x.Id == id);

if (condition is null)

{

return NotFound();

}

\_repository.Delete(condition);

var result = await \_repository.SaveChangesAsync();

if (result)

{

return NoContent();

}

return BadRequest();

}

2.4. Collect User Data

Mô tả: Tạo API có các input

* user id: truyền null nếu chưa có id, ngược lại truyền lên id user
* question id: Mã câu hỏi mà BOT đang hỏi user
* input: nội dung câu trả lời của user

[HttpPost]

public async Task<IActionResult> CollectUser(int? userId,

int questionId,

string input)

{

var question = Question.Items.FirstOrDefault(x => x.QuestionId == questionId);

if (question is null)

{

return NotFound();

}

if (ModelState.IsValid)

{

Candidate? candidate = null;

if (userId is null)

{

candidate = new Candidate();

\_repository.Add(candidate);

var addResult = await \_repository.SaveChangesAsync();

if (!addResult)

{

return BadRequest();

}

}

else

{

candidate = \_repository.Candidates.FirstOrDefault(x => x.Id == userId);

if(candidate is null)

{

return NotFound();

}

}

switch (questionId)

{

case 1:

candidate.Age = Convert.ToInt32(input);

break;

case 2:

candidate.HomeTown = input;

break;

case 3:

candidate.University = input;

break;

}

\_repository.Update(candidate);

var result = await \_repository.SaveChangesAsync();

if (result)

{

return Ok(candidate);

}

}

return BadRequest();

}

2.5. Xây dựng các hàm xử lý để kiểm tra điều kiện

2.5.1. Định nghĩa lớp IMatch

namespace FlowApp.Abstracts

{

public interface IMatch

{

bool IsMatch(string userData, string input);

}

public static class MatchFactory

{

public static IMatch? Create(string matchCode)

{

if(matchCode == Match.Is.Code)

{

return new IsOperate();

}

if(matchCode == Match.IsGreaterThan.Code)

{

return new IsGreaterThan();

}

return null;

}

}

}

2.5.2. Định nghĩa lớp FlowEngine

namespace FlowApp.Concrete

{

public class FlowEngine : IFlowEngine

{

private readonly DataContext \_context;

public FlowEngine(DataContext context)

{

\_context = context;

}

public bool Run(int questionId, Candidate candidate)

{

// load conditions

var conditions = \_context.Conditions.Where(x => x.QuestionId == questionId).ToList();

foreach(var condition in conditions)

{

var rule = Rule.GetRule(condition.TargetRule);

var match = MatchFactory.Create(condition.OperateMatch);

if (rule == Rule.Age)

{

if(!match.IsMatch(candidate.Age.ToString(), condition.Input))

{

return false;

}

continue;

}

if(rule == Rule.Location)

{

if (!match.IsMatch(candidate.HomeTown, condition.Input))

{

return false;

}

continue;

}

if(rule == Rule.University)

{

if (!match.IsMatch(candidate.University, condition.Input))

{

return false;

}

continue;

}

}

return true;

}

}

}