# Life Cycle

One Task Story

# BookLook

# New Query – Reference Point

*City...*

*Arrival Date...*

*Guests...*

*Deprature Date...*

?

# New Query – Reference Point .v1

*City...*

*Arrival Date...*

*Deprature Date...*

*Guests...*

*Reference Point Address...*

?

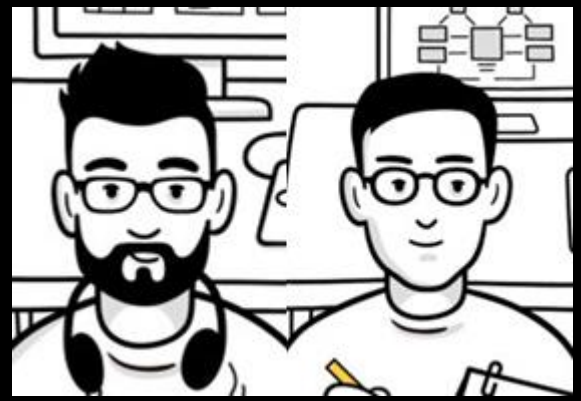
# New Query – Reference Point .v2

*City...*

*Arrival Date...*

*Deprature Date...*

*Guests...*



?

**Reference Point**

*City...*

*Street...*

*Building...*

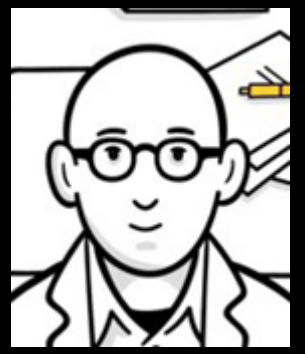
# New Query – Reference Point .v2

City...

Arrival Date...

Deprature Date...

Guests...



Ok

Reference Point

City...

Street...

Building...

# New Query – Reference Point .v2

City...

Arrival Date...

Deprature Date...

Guests...

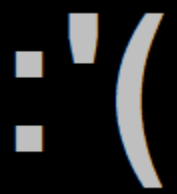
Reference Point

City...

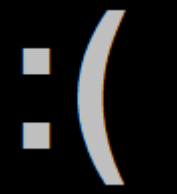
Street...

Building...

# New Query – Reference Point .v2



```
namespace WebApplication.Implementation.VirtualMachines
{
    public class VirtualMachinesService : IVirtualMachinesService
    {
        private readonly IVirtualMachinesRunspacePool virtualMachinesRunspacePool;
        private readonly IVirtualMachinesExecuteLog virtualMachinesExecuteLog;
        private static readonly Dictionary<Guid, CommandExecute> Executes = new Dictionary<Guid, CommandExecute>();

        public VirtualMachinesService(IVirtualMachinesRunspacePool virtualMachinesRunspacePool, IVirtualMachinesExecuteLog virtualMachinesExecuteLog)
        {
            this.virtualMachinesRunspacePool = virtualMachinesRunspacePool;
            this.virtualMachinesExecuteLog = virtualMachinesExecuteLog;
        }

        public Guid ExecuteCommandAsync(string command, string login, string[] virtualMachineNames)
        {
            var executeId = Guid.NewGuid();
            var powerShellResults = new Dictionary<string, AsyncPowershell>();
            foreach (var virtualMachine in virtualMachineNames)
            {
                var powerShell = PowerShell.Create();
                powerShell.Runspace = virtualMachinesRunspacePool.OpenRunspace(virtualMachine);
                powerShell.AddScript(command);
                powerShellResults[virtualMachine] = new AsyncPowershell(powerShell);
            }
            Executes.Add(executeId, new CommandExecute { Id = executeId, ProgressInfo = powerShellResults, Command = command, Login = login });

            return executeId;
        }

        public CommandExecuteResult GetExecuteCommandProgress(Guid executeId)
        {
            if (!Executes.ContainsKey(executeId))
            {
                return null;
            }

            var commandExecute = Executes[executeId];

            var machineResults = new Dictionary<string, string[]>();
            var machines = commandExecute.ProgressInfo.Select(x => x.Key).ToArray();
            foreach (var virtualMachine in machines)
            {
                var myPowershell = commandExecute.ProgressInfo[virtualMachine];
                if (!myPowershell.IsFinished())
                {
                    machineResults[virtualMachine] = new [] { "Is running", "" };
                    continue;
                }

                try
                {
                    machineResults[virtualMachine] = myPowershell.GetFormattedResult();
                }
                catch (Exception ex)
                {
                    machineResults[virtualMachine] = new [] { "", ex.Message };
                }
                finally
                {
                    myPowershell.Dispose();
                }
            }

            var result = new CommandExecuteResult
            {
                Id = commandExecute.Id,
                Command = commandExecute.Command,
                Login = commandExecute.Login,
                MachineResults = machines.Select(x => new CommandExecuteVirtualMachineResult
                {
                    MachineName = x,
                    ResultText = machineResults[x][0],
                    ErrorText = machineResults[x][1],
                    IsFinished = commandExecute.ProgressInfo[x].IsFinished(),
                    IsSuccess = commandExecute.ProgressInfo[x].IsSuccess()
                }).ToArray()
            };

            virtualMachinesExecuteLog.WriteLog(result);

            return result;
        }

        public void RefreshExecuteCommandProgress(Guid executeId)
        {
            GetExecuteCommandProgress(executeId);
        }
    }
}
```



# New Query – Reference Point .v3

**Reference Point**

***Country, Region, City, Street, Building...***

# New Query – Reference Point .v3

*City...*

*Arrival Date...*

*Deprature Date...*

*Guests...*

Ok

Reference Point

*Country, Region, City, Street, Building...*

# New Query – Reference Point .v3

```
public VerificationResult VerifyAddress(KladrAddress kladrAddress)
{
    // По кладру нет никаких корпусов, литер и строений, для удобства решил инкапсулировать здесь.
    var house = kladrAddress.House;
    var building = kladrAddress.Building;
    var room = kladrAddress.Room;

    var request = Request.Post("/v1/verify")
        .WithContent(new { address = kladrAddress }, serializer);
    var result = kladrServiceClient.Send<VerificationResult>(request);

    if (result.Address != null)
    {
        result.Address.House = house;
        result.Address.Building = building;
        result.Address.Room = room;
    }
    return result;
}
```

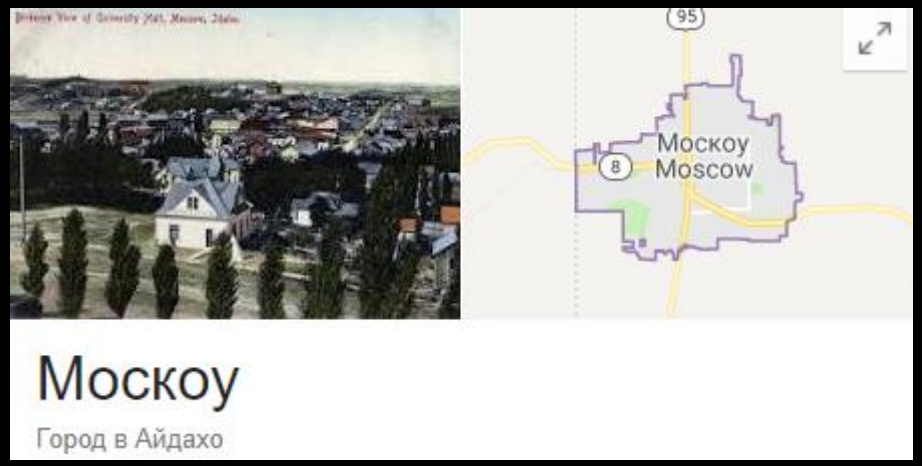
Ok

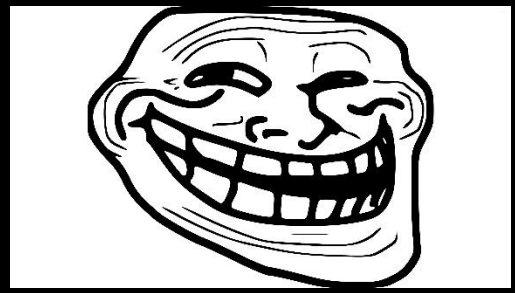
# New Query – Reference Point .v3

```
<div class="docflow-form-row">
    <span class="inlineBlock docflow-form-label">Регион</span>

    @Html.TextBox("regionCodeAutocomplete", Model.RegionCode, new Dictionary<string, object>
    {
        {"class", "textInput docflow-form-input js-form-input"},
        {"id", "DocflowDocumentInfoRefionInput"},
        { "data-url", Url.Action("GetRegions", "ClientRequisites") }
    })

    @Html.ValidationMessage("regionCodeAutocomplete", new Dictionary<string, object>
    {
        { "class", "updateRequisites-validation-error" }
    })
```

Ok

# New Query – Reference Point .v3



**Reference Point**

**USA, Moscow**

:(

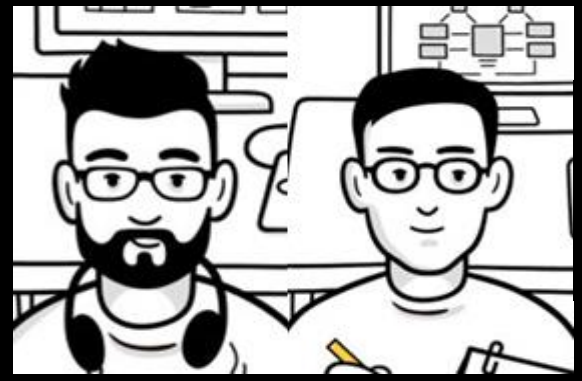
# New Query – Reference Point .v4

**Reference Point**

***Moscow...***

**Russia, Moscow**

**USA, Moscow**



?

Ok

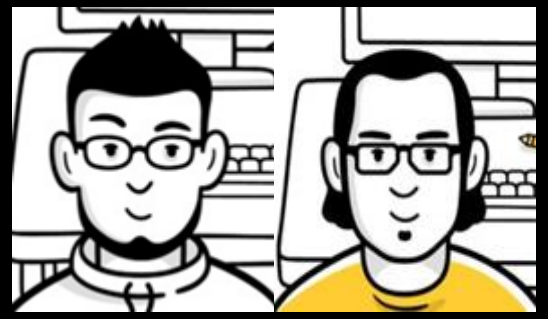
# New Query – Reference Point .v4

**Reference Point**

***Moscow...***

Russia, Moscow

USA, Moscow



Ok

# New Query – Retrospective