Troisième séance

28-11-2015

05-12-2015

Agenda

- Transactions
- Architecture de votre projet
- Travail sur votre projet

Transactions

- Suivez les consignes publiées sur Claroline

Transactions: mise en commun

- Critique de l’exemple proposé et de sa problématique

Architecture de votre projet

- Beaucoup de possibilités
- Deux approches vous sont proposées pour votre projet de base de données:
  - Single Page Application (SPA) avec AngularJS et Web API
  - Application ASP.NET MVC avec Razor

Deux approches vous sont proposées

SPA : AngularJS + Web API

- Application composite
  - Shell = Page (= P de SPA)
  - Vues (fragments HTML) affichées dans le Shell
- Un peu comme les Frames HTML
- On vise la performance :
  - pas de full page reload
  - génération de markup (listes etc…): côté client, on soulage le serveur
- Vous devez connaître le HTML et le Javascript (ou vouloir l'apprendre)
- Pattern MVC/MVVM : MVW (Model-View-Whatever): https://plus.google.com/+AngularJS/posts/aZNVhj355G2

ASP.NET MVC + Razor

- Vues HTML générées côté serveur (principe similaire à PHP)
- Vues décrites à l'aide d'un mélange de HTML et c# à l'aide de Razor
- Full page reloads fréquents, même si Ajax possible.
- Vous devez connaître C# et HTML
- Pattern MVC

Comparaison des approches pour une vue affichant une liste d'éléments

SPA : AngularJS + Web API: le markup sera généré côté client (dans le navigateur)

```
<h1>Liste des utilisateurs</h1>

<table class="table">
    <thead>
        <tr>
            <th>Nom</th>
            <th>E-mail</th>
            <th>Roles</th>
            <th>Statut</th>
        </tr>
    </thead>
    <tr ng-repeat="user in users">

        <td><a href="#/security/users/details/{{user.Id}}">{{ user.UserName }}</a></td>
        <td><a href="#/security/users/details/{{user.Id}}">{{ user.EMail }}</a></td>
        <td width="200"><span ng-repeat="role in user.Roles">{{role.Name}}{{$last ? '' : ', '}}
        <td width="200">{{user.Status}}</td>
    </tr>
</table>
```

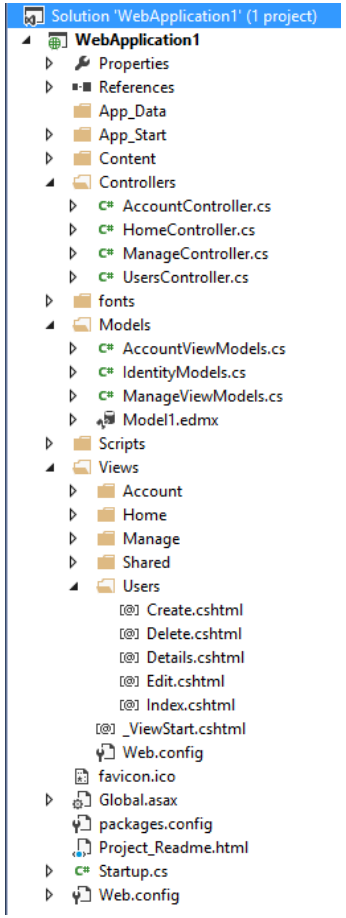
ASP.NET MVC + Razor: ce code génèrera du markup (côté serveur)

```
@model IEnumerable<WebApplication1.Models.AspNetUser>

@{
    ViewBag.Title = "Index";
}
<h1>Liste des utilisateurs</h1>

<table class="table">
    <thead>
        <tr>
            <th>Nom</th>
            <th>E-Mail</th>
            <th>Roles</th>
            <th>Statut</th>
        </tr>
    </thead>
    @foreach (var user in Model)
    {
        <tr>
            <td>
                @Html.ActionLink(user.UserName, "Details", new { id = user.Id })
            </td>
            <td>
                @Html.ActionLink(user.Email, "Details", new { id = user.Id })
            </td>
            <td>
                @string.Join(", ", user.AspNetRoles.Select(role => role.Name).ToArray())
            </td>
            <td>
                @user.Status
            </td>
        </tr>
    }
</table>
```

Structure d'une solution ASP.NET MVC + Razor



- Model
  - Logique de votre application
  - Classes C#
- Controller:
  - Classes C#
  - De type Controller (pas ApiController!)
- View:
  - Fichiers .cshtml (HTML contenant du C# déclaré à l'aide de **Razor**)
  - 1 folder par Controller (concordance des noms, convention)
  - N views par Controller

Structure d'une solution ASP.NET MVC + Razor

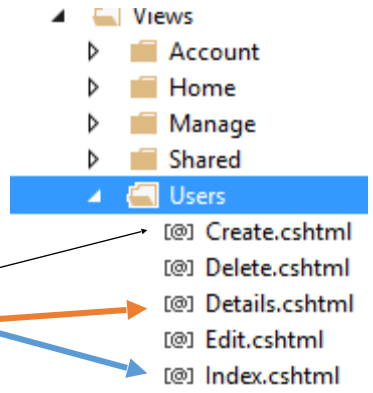
Controller

```
public class UsersController : Controller
{
    private Entities db = new Entities();
                          Entities.Entities()
    // GET: Users
    public ActionResult Index()
    {
        return View(db.AspNetUsers.Include(u => u.AspNetRoles).ToList());
    }

    // GET: Users/Details/5
    public ActionResult Details(string id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        AspNetUser aspNetUser = db.AspNetUsers.Find(id);
        if (aspNetUser == null)
        {
            return HttpNotFound();
        }
        return View(aspNetUser);
    }

    // GET: Users/Create
    public ActionResult Create()
    {
        return View();
```

Vues



En réponse à une requête HTTP reçue du client, le controller renvoie la vue correspondante (convention sur base du nom de l'action qui doit correspondre au nom de la vue) en définissant le modèle de celle-ci (passé en paramètre à la méthode View()).

ASP.NET MVC + Razor: exemple d'une Vue

```
@model IEnumerable<WebApplication1.Models.AspNetUser>

@{
    ViewBag.Title = "Index";
}
<h1>Liste des utilisateurs</h1>

<table class="table">
    <thead>
        <tr>
            <th>Nom</th>
            <th>E-Mail</th>
            <th>Roles</th>
            <th>Statut</th>
        </tr>
    </thead>
    @foreach (var user in Model)
    {
        <tr>
            <td>
                @Html.ActionLink(user.UserName, "Details", new { id = user.Id })
            </td>
            <td>
                @Html.ActionLink(user.Email, "Details", new { id = user.Id })
            </td>
            <td>
                @string.Join(", ", user.AspNetRoles.Select(role => role.Name).ToArray())
            </td>
            <td>
                @user.Status
            </td>
        </tr>
    }
</table>
```

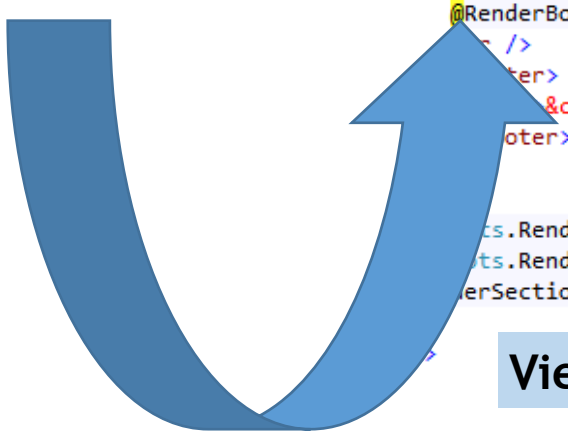
- La vue est définie en Razor
- Le serveur va parser la vue et générer du HTML sur base des valeurs du modèle lié à la vue (c'est le controller qui lie le modèle à la vue lors de l'appel à View()).
- Ce n'est pas un document HTML complet, c'est un fragment de HTML, un « View Template ».
- Ce fragment va être inclus à un document HTML commun (Shared _Layout.cshtml) lorsque ce dernier appelera la méthode RenderBody().
- Toutes ces opérations se passent côté serveur
- Le HTML résultant est retourné au client

Views/Users/Index.cshtml

```
@model IEnumerable<WebApplication1.Models.AspNetUser>

@{
    ViewBag.Title = "Index";
}
<h1>Liste des utilisateurs</h1>

<table class="table">
    <thead>
        <tr>
            <th>Nom</th>
            <th>E-Mail</th>
            <th>Roles</th>
            <th>Statut</th>
        </tr>
    </thead>
    @foreach (var user in Model)
    {
        <tr>
            <td>
                @Html.ActionLink(user.UserName, "Details", new { id = user.Id })
            </td>
            <td>
                @Html.ActionLink(user.Email, "Details", new { id = user.Id })
            </td>
            <td>
                @string.Join(", ", user.AspNetRoles.Select(role => role.Name).ToArray())
            </td>
            <td>
                @user.Status
            </td>
        </tr>
    }
</table>
```



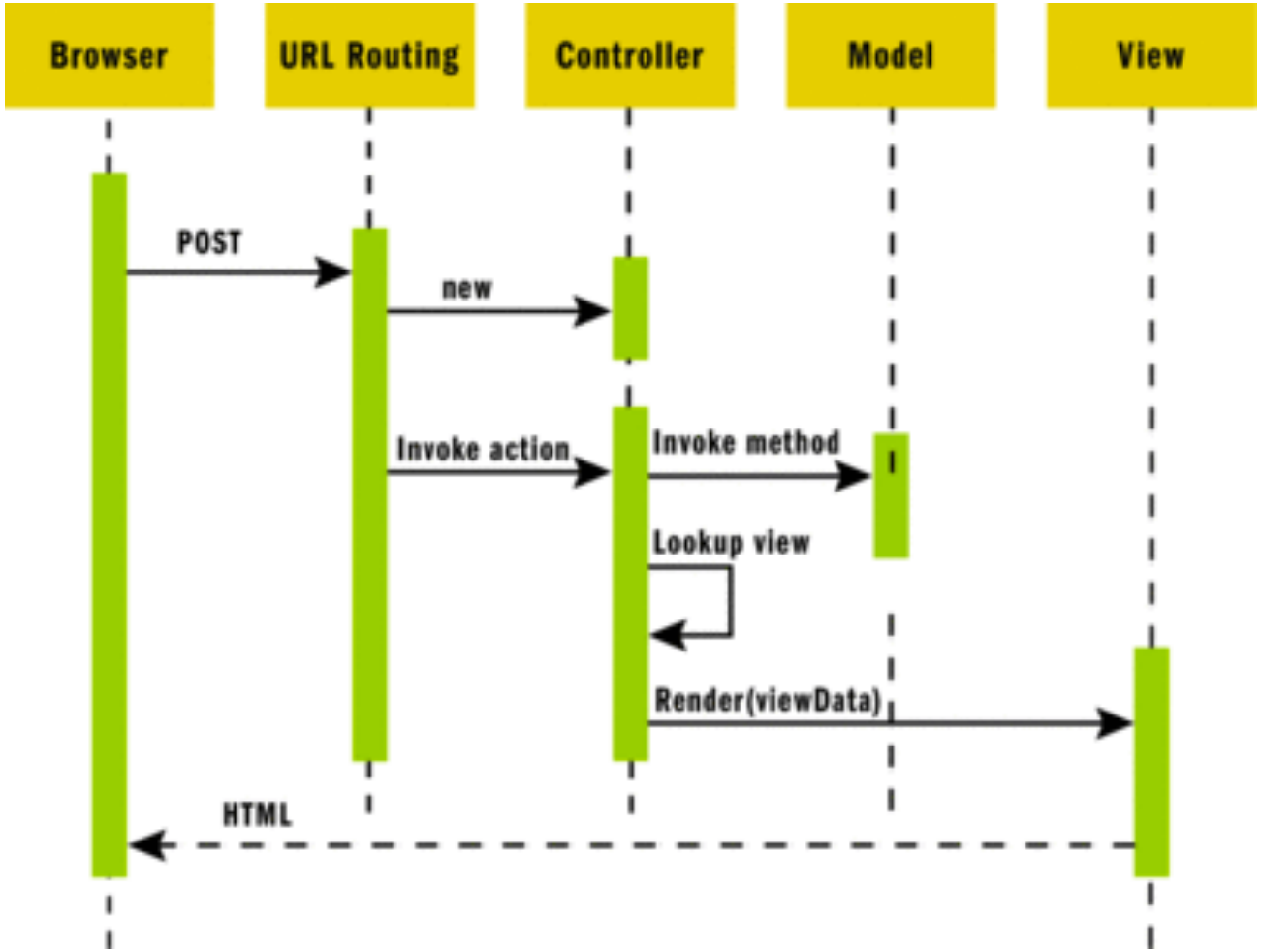
Views/Shared/_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        - />
        ter>
            &copy; @DateTime.Now.Year - My ASP.NET Application</p>
        oter>


        ts.Render("~/bundles/jquery")
        ts.Render("~/bundles/bootstrap")
        erSection("scripts", required: false)
```

ASP.NET MVC + Razor: séquence

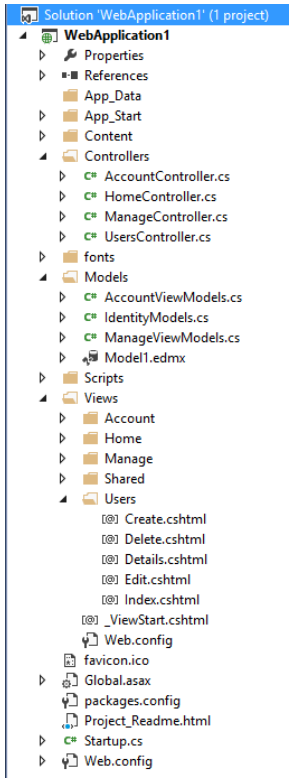


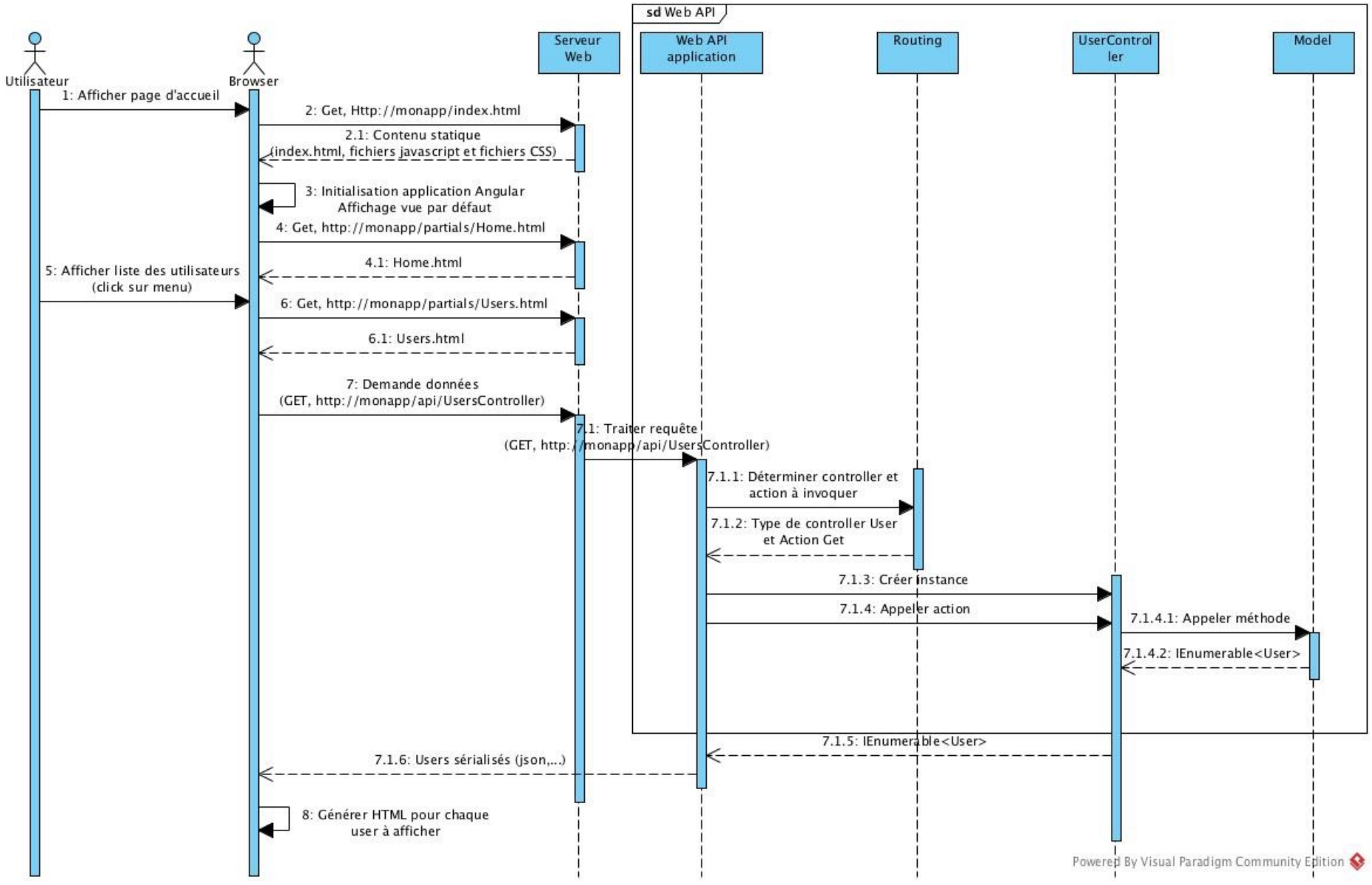
https://msdn.microsoft.com/en-us/magazine/dd942833.aspx

Structure d’une solution Angular JS + Web API

SPA : AngularJS + Web API

ASP.NET MVC + Razor





Version simplifiée. Détail complet du traitement d'une requête disponible sur http://www.microsoft.com/en-in/download/confirmation.aspx?id=36476

Quelle approche devez-vous choisir?

- Question ouverte. Exemple d'arguments: http://programmers.stackexchange.com/questions/238647/pure-front-end-javascript-with-web-api-versus-mvc-views-with-ajax
- Autres arguments:
  - La première approche
    - Est du point de vue des performances la meilleure (serveur « soulagé »)
    - Va vous forcer à créer une Web API qui peut être réutilisée pour d'autres clients que le client Web que vous allez créer.
    - Respecte plus le principe de séparation des responsabilités
  - La seconde approche
    - Vous permet de bénéficier des avantages de la compilation pour détecter certaines erreurs avant l'exécution (en Javascript, vous ne constateriez les erreurs qu'à l'exécution)
    - Limite le Javascript que vous devrez apprendre
    - Est plus orientée RAD (Rapid Application Development). Vous serez de ce fait plus bridé qu'avec la première approche.
- Tout est à remettre en contexte:
  - On ne vous demande dans ce projet qu'un seul client: le client Web
  - Votre stage à venir et les technologies impliquées
  - Votre envie de vous améliorer en Javascript ou pas
- Faites votre choix et sachez le justifier

Les deux approches peuvent-elles être mélangées?

- Oui. Exemple d'un descriptif: https://newsignature.com/articles/modern-front-end-development-for-net-with-angularjs
- Mais cela vous est déconseillé pour ce projet (afin de garder les idées claires).
- Néanmoins, si vous optez pour l'approche SPA + ASP.NET Web API, rien ne vous empêche de tout héberger dans un même projet Visual Studio de type Web Application.

Ressources

SPA: AngularJS + Web API

- https://docs.angularjs.org/misc/started
- http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/build-a-single-page-application-spa-with-aspnet-web-api-and-angularjs
- http://www.johnpapa.net/pageinspa/
- Exemple d'application: http://cc-ng-z.azurewebsites.net/#/

ASP.NET MVC Razor

- http://www.asp.net/web-pages/overview/getting-started/introducing-razor-syntax-c

Exercices

- Réalisez une première application, se basant sur ASP.NET MVC
- Réalisez une seconde application, se basant sur ASP.NET Web API + AngularJS
- Voir consignes sur Claroline (ConsignesLaboratoireChoixArchitecture.pdf)