|  |
| --- |
| **BTS Services informatiques aux organisations SESSION 2024**  **ANNEXE 9-1-B : Fiche descriptive de réalisation professionnelle (recto)**  **Épreuve E5 - Conception et développement d’applications (option SLAM)** |

|  |  |  |  |
| --- | --- | --- | --- |
| **DESCRIPTION D’UNE RÉALISATION PROFESSIONNELLE** | | | **N° réalisation :** |
| **Nom, prénom : Vadenne Johan** | | **N° candidat :** | |
| **Épreuve ponctuelle** | **Contrôle en cours de formation** | **Date :** ...... / ...... /............ | |
| **Organisation support de la réalisation professionnelle**  Entreprise VAD (Visual Application Designer) | | | |
| **Intitulé de la réalisation professionnelle**  Application WEB (HTML, CSS, TWIG, PHP, Symfony) : Gestion des stages & apprentissages | | | |
| **Période de réalisation :** 04/03/2024 – 23/03/2024 **Lieu :** La Rochelle  **Modalité :  Seul(e)  En équipe** | | | |
| **Compétences travaillées**  Concevoir et développer une solution applicative  Assurer la maintenance corrective ou évolutive d’une solution applicative  Gérer les données | | | |
| **Conditions de réalisation[[1]](#footnote-1) (ressources fournies, résultats attendus)**  **Description** : Gestion des stages & apprentissages est une application web conçue pour administrer les étudiants en stage ou en apprentissage. Cette application facilite la création de profils étudiants et leur association à des formations spécifiques. Une fois qu'un stage est trouvé, il peut être ajouté et lié à l'entreprise correspondante. La base de données répertorie toutes les entreprises enregistrées par les professeurs  **Objectif** :  L'objectif de cette application est de permettre aux professeurs du lycée Fénelon Notre Dame de gérer plus facilement les stages et les apprentissages des étudiants. Ainsi, la liste des entreprises enregistrées dans la base de données peut aider les étudiants à trouver un stage ou un apprentissage. | | | |
| **Description des ressources documentaires, matérielles et logicielles utilisées[[2]](#footnote-2)**  Ordinateur :  PC fixe Fénelon Notre Dame  VM Debian Fénelon Notre Dame  PC portable Asus Personnelle  Logiciel :  FireFox  Visual Studio  Symfony  Looping  draw io  DB Browser (SQLite)  Composer:  "php": ">=8.2",  "ext-ctype": "\*",  "ext-iconv": "\*",  "doctrine/dbal": "^3",  "doctrine/doctrine-bundle": "^2.12",  "doctrine/doctrine-migrations-bundle": "^3.3",  "doctrine/orm": "^3.1",  "symfony/asset": "7.0.\*",  "symfony/console": "7.0.\*",  "symfony/dotenv": "7.0.\*",  "symfony/flex": "^2",  "symfony/form": "7.0.\*",  "symfony/framework-bundle": "7.0.\*",  "symfony/monolog-bundle": "^3.10",  "symfony/runtime": "7.0.\*",  "symfony/security-bundle": "7.0.\*",  "symfony/security-csrf": "7.0.\*",  "symfony/twig-bundle": "7.0.\*",  "symfony/validator": "7.0.\*",  "symfony/yaml": "7.0.\*"  "phpunit/phpunit": "^9.5",  "symfony/browser-kit": "7.0.\*",  "symfony/css-selector": "7.0.\*",  "symfony/maker-bundle": "\*",  "symfony/phpunit-bridge": "^7.0"  Documentation :  <https://chat.openai.com/>  <https://symfony.com> | | | |
| **Modalités d’accès aux productions[[3]](#footnote-3) et à leur documentation[[4]](#footnote-4)** UML BDD  UML Classe  Schéma environnement informatique   Fenêtre   SécuritéLe Hash "auto" de Symfony Il sélectionne automatiquement le meilleur hachoir disponible (actuellement Bcrypt). Si PHP ou Symfony ajoute de nouveaux hachages de mots de passe dans le futur, il pourrait sélectionner un hachoir différent.  Pour cette raison, la longueur des mots de passe hachés peut changer dans le futur, donc il faut s’assurer d'allouer assez d'espace pour qu'ils soient persistants (varchar(255) devrait être un bon réglage). Le Bcrypt Password Hasher Il produit des mots de passe hachés avec la fonction de hachage de mot de passe de bcrypt. Les mots de passe hachés ont une longueur de 60 caractères, assurez-vous donc d'allouer suffisamment d'espace pour qu'ils soient conservés. De plus, les mots de passe contiennent le sel cryptographique (il est généré automatiquement pour chaque nouveau mot de passe), vous n'avez donc pas à vous en préoccuper.  Sa seule option de configuration est le coût, qui est un entier compris entre 4 et 31 (par défaut, 13). Chaque incrément du coût double le temps nécessaire pour hacher un mot de passe. Il est conçu de cette manière afin que la force du mot de passe puisse être adaptée aux améliorations futures de la puissance de calcul.  Vous pouvez modifier le coût à tout moment, même si certains mots de passe ont déjà été hachés à l'aide d'un coût différent. Les nouveaux mots de passe seront hachés en utilisant le nouveau coût, tandis que les mots de passe déjà hachés seront validés en utilisant le coût qui a été utilisé lors de leur hachage.  wikipédia: <https://fr.wikipedia.org/wiki/Bcrypt> <https://symfony.com/doc/current/security/passwords.html#reference-security-encoder-bcrypt> BDD Toutes les manipulations effectuées avec la base de données sont gérées par l'ORM Doctrine.  <https://symfony.com/doc/current/doctrine.html>  <https://fr.wikipedia.org/wiki/Doctrine_(ORM)> | | | |

|  |
| --- |
| AuthenticationFenêtresecurity.yaml     access\_control:          - { path: ^/register, roles: [ROLE\_ADMIN] }          - { path: ^/etudiant, roles: [ROLE\_ADMIN,ROLE\_PROF] }          - { path: ^/etude, roles: [ROLE\_ADMIN,ROLE\_PROF] }          - { path: ^/entreprise, roles: [ROLE\_ADMIN,ROLE\_PROF] }          - { path: ^/stage/apprentissage, roles: [ROLE\_ADMIN,ROLE\_PROF] } SecurityController.php <?php  namespace App\Controller;  class SecurityController extends AbstractController  {      private $logger;      public function \_\_construct(LoggerInterface $logger)      {          $this->logger = $logger;      }      #[Route(path: '/login', name: 'app\_login')]      public function login(AuthenticationUtils $authenticationUtils): Response      {          // if ($this->getUser()) {          //     return $this->redirectToRoute('target\_path');          // }          // get the login error if there is one          $error = $authenticationUtils->getLastAuthenticationError();          // last username entered by the user          $lastUsername = $authenticationUtils->getLastUsername();          if ($error) {              $username = $lastUsername ?? 'Unknown';              $ip = $this->container->get('request\_stack')->getCurrentRequest()->getClientIp();              $this->logger->error(sprintf(                  'Tentative de connexion échouée - Utilisateur: %s, IP: %s, Erreur: %s',                  $username,                  $ip,                  $error->getMessage()              )); // log          }          return $this->render('security/login.html.twig', ['last\_username' => $lastUsername, 'error' => $error]);      }      #[Route(path: '/logout', name: 'app\_logout')]      public function logout(): void      {          throw new \LogicException('This method can be blank - it will be intercepted by the logout key on your firewall.');      }  } User.php <?php  namespace App\Entity;  #[ORM\Entity(repositoryClass: UserRepository::class)]  #[ORM\UniqueConstraint(name: 'UNIQ\_IDENTIFIER\_USERNAME', fields: ['username'])]  #[UniqueEntity(fields: ['username'], message: 'There is already an account with this username')]  class User implements UserInterface, PasswordAuthenticatedUserInterface  {      #[ORM\Id]      #[ORM\GeneratedValue]      #[ORM\Column]      private ?int $id = null;      #[ORM\Column(length: 180)]      private ?string $username = null;      /\*\*       \* @var list<string> The user roles       \*/      #[ORM\Column]      private array $roles = [];      /\*\*       \* @var string The hashed password       \*/      #[ORM\Column]      private ?string $password = null;      public function getId(): ?int      {          return $this->id;      }      public function getUsername(): ?string      {          return $this->username;      }      public function setUsername(string $username): static      {          $this->username = $username;          return $this;      }      /\*\*       \* A visual identifier that represents this user.       \*       \* @see UserInterface       \*/      public function getUserIdentifier(): string      {          return (string) $this->username;      }      /\*\*       \* @see UserInterface       \*       \* @return list<string>       \*/      public function getRoles(): array      {          $roles = $this->roles;          // guarantee every user at least has ROLE\_USER          $roles[] = 'ROLE\_USER';          return array\_unique($roles);      }      /\*\*       \* @param list<string> $roles       \*/      public function setRoles(array $roles): static      {          $this->roles = $roles;          return $this;      }      /\*\*       \* @see PasswordAuthenticatedUserInterface       \*/      public function getPassword(): string      {          return $this->password;      }      public function setPassword(string $password): static      {          $this->password = $password;          return $this;      }      /\*\*       \* @see UserInterface       \*/      public function eraseCredentials(): void      {          // If you store any temporary, sensitive data on the user, clear it here          // $this->plainPassword = null;      }  } UserRepository.php <?php  namespace App\Repository;  /\*\*   \* @extends ServiceEntityRepository<User>   \*   \* @method User|null find($id, $lockMode = null, $lockVersion = null)   \* @method User|null findOneBy(array $criteria, array $orderBy = null)   \* @method User[]    findAll()   \* @method User[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)   \*/  class UserRepository extends ServiceEntityRepository implements PasswordUpgraderInterface  {      public function \_\_construct(ManagerRegistry $registry)      {          parent::\_\_construct($registry, User::class);      }      /\*\*       \* Used to upgrade (rehash) the user's password automatically over time.       \*/      public function upgradePassword(PasswordAuthenticatedUserInterface $user, string $newHashedPassword): void      {          if (!$user instanceof User) {              throw new UnsupportedUserException(sprintf('Instances of "%s" are not supported.', $user::class));          }          $user->setPassword($newHashedPassword);          $this->getEntityManager()->persist($user);          $this->getEntityManager()->flush();      }  } GestionEtudiantStageAuthenticator.php <?php  namespace App\Security;  class GestionEtudiantStageAuthenticator extends AbstractLoginFormAuthenticator  {      use TargetPathTrait;      public const LOGIN\_ROUTE = 'app\_login';      private $logger;      public function \_\_construct(private UrlGeneratorInterface $urlGenerator, LoggerInterface $logger)      {          $this->logger = $logger;      }      public function authenticate(Request $request): Passport      {          $username = $request->getPayload()->getString('username');          $request->getSession()->set(SecurityRequestAttributes::LAST\_USERNAME, $username);          return new Passport(              new UserBadge($username),              new PasswordCredentials($request->getPayload()->getString('password')),              [                  new CsrfTokenBadge('authenticate', $request->getPayload()->getString('\_csrf\_token')),                  new RememberMeBadge(),              ]          );      }      public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response      {          if ($targetPath = $this->getTargetPath($request->getSession(), $firewallName)) {              $session = $request->getSession();              $lastUsername = $session->get('\_security.last\_username');              $username = $lastUsername ?? 'Unknown';              $ip = $request->getClientIp();              $this->logger->info(sprintf('Connexion réussie - Utilisateur: %s, IP: %s', $username, $ip)); // log              return new RedirectResponse($targetPath);          }          // For example:          // return new RedirectResponse($this->urlGenerator->generate('some\_route'));          throw new \Exception('TODO: provide a valid redirect inside '.\_\_FILE\_\_);      }      protected function getLoginUrl(Request $request): string      {          return $this->urlGenerator->generate(self::LOGIN\_ROUTE);      }  } login.html.twig {% extends 'base.html.twig' %}  {% block title %}Log in!{% endblock %}  {% block body %}  <form method="post">      {% if error %}          <div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>      {% endif %}      {% if app.user %}          <div class="mb-3">              You are logged in as {{ app.user.userIdentifier }}, <a href="{{ path('app\_logout') }}">Logout</a>          </div>      {% endif %}      <h1 class="h3 mb-3 font-weight-normal">Connexion</h1>      <label for="inputUsername">Utilisateur</label>      <input type="text" value="{{ last\_username }}" name="username" id="inputUsername" class="form-control" autocomplete="username" required autofocus>      <label for="inputPassword">Mot de passe</label>      <input type="password" name="password" id="inputPassword" class="form-control" autocomplete="current-password" required>      <input type="hidden" name="\_csrf\_token"             value="{{ csrf\_token('authenticate') }}"      >      <div class="checkbox mb-3">          <label>              <input type="checkbox" name="\_remember\_me"> Se souvenir de moi          </label>      </div>      <button class="btn btn-lg btn-primary" type="submit">          Connexion      </button>  </form>  {% endblock %} Ajout d’un nouvel étudiantFenêtre    EtudiantController.php <?php  namespace App\Controller;  #[Route('/etudiant')]  class EtudiantController extends AbstractController  {      private $logger;      public function \_\_construct(LoggerInterface $logger)      {          $this->logger = $logger;      }        #[Route('/', name: 'app\_etudiant\_index', methods: ['GET'])]      public function index(EtudiantRepository $etudiantRepository): Response      {          return $this->render('etudiant/index.html.twig', [              'etudiants' => $etudiantRepository->findAllWithRelationEtude(),          ]);      }      #[Route('/new', name: 'app\_etudiant\_new', methods: ['GET', 'POST'])]      public function new(Request $request, EntityManagerInterface $entityManager): Response      {          $etudiant = new Etudiant();          $form = $this->createForm(EtudiantType::class, $etudiant);          $form->handleRequest($request);          if ($form->isSubmitted() && $form->isValid()) {              $this->logger->info('NOUVELLE ÉTUDIANT'); // log              $entityManager->persist($etudiant);              $entityManager->flush();              return $this->redirectToRoute('app\_etudiant\_index', [], Response::HTTP\_SEE\_OTHER);          }          return $this->render('etudiant/new.html.twig', [              'etudiant' => $etudiant,              'form' => $form,          ]);      }      #[Route('/{id}', name: 'app\_etudiant\_show', methods: ['GET'])]      public function show(Etudiant $etudiant): Response      {          return $this->render('etudiant/show.html.twig', [              'etudiant' => $etudiant,          ]);      }  } Etudiant.php <?php  namespace App\Entity;  #[ORM\Entity(repositoryClass: EtudiantRepository::class)]  class Etudiant  {      #[ORM\Id]      #[ORM\GeneratedValue]      #[ORM\Column]      private ?int $id = null;      #[ORM\Column(length: 50)]      private ?string $NomEleve = null;      #[ORM\Column(length: 50)]      private ?string $PrenomEtudiant = null;      #[ORM\ManyToOne(inversedBy: 'etudiants')]      private ?Etude $IdEtude = null;      #[ORM\OneToMany(targetEntity: StageApprentissage::class, mappedBy: 'IdEtudiant', orphanRemoval: true)]      private Collection $stageApprentissages;      public function \_\_construct()      {          $this->stageApprentissages = new ArrayCollection();      }      public function getId(): ?int      {          return $this->id;      }      public function getNomEleve(): ?string      {          return $this->NomEleve;      }      public function setNomEleve(string $NomEleve): static      {          $this->NomEleve = $NomEleve;          return $this;      }      public function getPrenomEtudiant(): ?string      {          return $this->PrenomEtudiant;      }      public function setPrenomEtudiant(string $PrenomEtudiant): static      {          $this->PrenomEtudiant = $PrenomEtudiant;          return $this;      }      public function getIdEtude(): ?Etude      {          return $this->IdEtude;      }      public function setIdEtude(?Etude $IdEtude): static      {          $this->IdEtude = $IdEtude;          return $this;      }  } EtudiantType.php <?php  namespace App\Form;  class EtudiantType extends AbstractType  {      public function buildForm(FormBuilderInterface $builder, array $options): void      {          $builder              ->add('NomEleve')              ->add('PrenomEtudiant')              ->add('IdEtude', EntityType::class, [                  'class' => Etude::class,                  'choice\_label' => 'NomEtude',              ])          ;      }      public function configureOptions(OptionsResolver $resolver): void      {          $resolver->setDefaults([              'data\_class' => Etudiant::class,          ]);      }  } EtudiantRepository.php <?php  namespace App\Repository;  use App\Entity\Etudiant;  use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;  use Doctrine\Persistence\ManagerRegistry;  class EtudiantRepository extends ServiceEntityRepository  {      public function \_\_construct(ManagerRegistry $registry)      {          parent::\_\_construct($registry, Etudiant::class);      }      // Permets de faire une jointure avec l'entité Étude      public function findAllWithRelationEtude()      {          return $this->createQueryBuilder('e')              ->leftJoin('e.IdEtude', 'ie')              ->addSelect('ie')              ->getQuery()              ->getResult();      }  } Etudiant: new.html.twig {% extends 'base.html.twig' %}  {% block title %}Nouvelle étudiant{% endblock %}  {% block body %}      <h1>Nouvelle étudiant</h1>      {{ include('etudiant/\_form.html.twig') }}      <a class="btn btn-secondary" href="{{ path('app\_etudiant\_index') }}">retour</a>  {% endblock %} |

|  |
| --- |
| **BTS Services informatiques aux organisations SESSION 2024**  **ANNEXE 9-1-B : Fiche descriptive de réalisation professionnelle**  **(verso, éventuellement pages suivantes)**  **Épreuve E5 - Conception et développement d’applications (option SLAM)** |

|  |
| --- |
|  |

1. En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO. [↑](#footnote-ref-1)
2. Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l’annexe II.E du référentiel du BTS SIO. [↑](#footnote-ref-2)
3. Conformément au référentiel du BTS SIO « *Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l’épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d’organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l’épreuve.* ». Les éléments peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d’un espace de stockage et de la présentation de l’organisation du stockage. [↑](#footnote-ref-3)
4. Lien vers la documentation complète, précisant et décrivant, si cela n’a été fait au verso de la fiche, la réalisation professionnelle, par exemples service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données. [↑](#footnote-ref-4)