Desarrollo E Implementación Del Sistema De Información Para El Banco De Hojas De Vida De La Universidad De Cundinamarca Utilizando Metodologías Ágiles De Desarrollo

Desarrollo de aplicaciones libres

ALUMNO: José Fernando Sotelo Cubillos <jfsotelo2000@yahoo.es>
TUTOR OUC: Gregorio Robles Martínez <grobles@uoc.edu>
TUTOR EXTERNO: Gustavo castillo <casgu@msm.com>

Resumen

El presente proyecto tiene como objetivo desarrollar e implementar el software para la gestión de hojas de vida de los profesores de la Universidad de Cundinamarca, mediante el cual se podrá gestionar y tener información de cada uno de los docentes tanto antiguos como nuevos que laboran dentro de cada una de las facultades de la universidad.

La metodología de desarrollo que se utilizó para el desarrollo del software es XP (programación extrema) la cual es una de las más utilizadas en la ingeniería de software donde atreves de desarrollo ágil se logra construir software en compañía del cliente quien es uno de los actores principales.

Se hizo un levantamiento de información y análisis de requisitos, donde se determinaron las principales necesidades que debía cubrir la aplicación, además se diseñó la base de datos y la interfaz gráfica necesaria para el desarrollo del software.

Se concluye que con el uso de este software la Universidad de Cundinamarca podrá obtener información oportuna de los docentes y por lo tanto podrá tomar decisiones en cuanto a la asignación de carga académica según el perfil y demás información de cada uno de ellos.

Finalmente en la realización de proyecto se han aplicado muchos de los conceptos teóricos tratados en el transcurso del master de software libre.

Abstract

This project aims to develop and implement software for managing resumes of teachers at the University of Cundinamarca, through which they can manage to have information on each of both old and new teachers who work within each one of the university faculties.

Development methodology that was used for software development is XP (extreme programming) which is one of the most used in software engineering where you dare agile development is achieved build software in the company of the end user who is one of the main actors.

Gathering information and requirements analysis, where the main needs that should cover the application was determined, plus the database and the necessary graphical interface for software development it was designed.

It is concluded that the use of this software the University of Cundinamarca can obtain timely information for teachers and therefore can make decisions regarding the allocation of academic load to the profile and other information in each.

Finally in conducting project they have implemented many of the theoretical concepts discussed during the master of free software.

Contenido

1. INTRODUCCIÓN

Este proyecto tiene la finalidad de desarrollar e implementar el sistema de información para el banco de hojas de vida de la universidad de Cundinamarca utilizando metodologías ágiles de desarrollo y su objetivo es gestionar las hojas de vida de los docentes que trabajan en la universidad y así poder tomar decisiones de asignación de asignaturas dependiendo del perfil del docente.

La tecnología escogida para hacer el desarrollo es Java, el entorno de desarrollo Netbeans y como motor de la base de datos Postgresql.

El diseño de la aplicación se ha hecho utilizando la metodología de desarrollo de software XP el cual consiste en tener en cuenta las necesidades de los usuarios en todas las fases del proceso de diseño.

1.1 Justificación

Con este proyecto se busca dar una solución óptima a una necesidad que existe dentro de la universidad, que es tener la información general de cada docente tanto nuevo como antiguo y poder tomar las decisiones necesarias tanto para su contratación como para actividades de asignación de carga, de investigación, de desarrollo curricular, y otras actividades propias de los docentes de la Universidad de Cundinamarca.

En cuanto al desarrollo de la aplicación se hará utilizando temas vistos durante el Master de Software Libre como lo son Bases de datos, diseño web, ingeniería de software entre otros.

Disponer de una dirección de correo electrónico, de acceso a la web, etc., ha dejado de ser una novedad para convertirse en algo normal en muchos países del mundo. Por eso las empresas, instituciones, administraciones y demás están migrando rápidamente todos sus servicios, aplicaciones, tiendas, etc., a un entorno web que permita a sus clientes y usuarios acceder a todo ello por Internet.

La WWW (World Wide Web) o, de forma más coloquial, la web, se ha convertido, junto con el correo electrónico, en el principal servicio de Internet. Ésta ha dejado de ser una inmensa “biblioteca” de páginas estáticas para convertirse en un servicio que permite acceder a multitud de prestaciones y funciones, así como a infinidad de servicios, programas, tiendas, etc. (Piñol)

El aplicativo desarrollado, permitirá a los docentes de la Universidad de Cundinamarca (UDEC) gestionar su hoja de vida, ingresando la información requerida, tanto sus datos básicos como su formación académica, cursos de actualización, seminarios, experiencia laboral, experiencia docente, etc.

El objetivo general de este trabajo es poner en práctica los conocimientos y habilidades adquiridos para planificar y desarrollar un proyecto de software. Dentro del contenido de algunas asignaturas de la maestría como lo son desarrollo web, desarrollo de software, ingeniería de software se han tratado el manejo de herramientas de uso libre para el desarrollo de proyectos de software.

2. OBJETIVO GENERAL

Desarrollar e implementar un sistema de información para el banco de hojas de vida de la universidad de Cundinamarca utilizando metodologías ágiles de desarrollo.

2.1 Objetivos Específicos

1. Consultar acerca de las metodologías agiles de desarrollo para la aplicación en el proyecto.
2. Diseñar e implantar el modelo de base de datos del aplicativo, definiendo los perfiles de usuario, para que realice operaciones en la base de acuerdo a su perfil
3. Diseñar las iteraciones siempre en compañía del usuario final, para descubrir los requerimientos del sistema.
4. Desarrollar las tareas planeadas en cada una de las iteraciones
5. Determinar el diseño lógico del sistema de información
6. Desarrollar la interfaz gráfica del sistema de información utilizando software libre.

2.2 Planificación

El cronograma de actividades a seguir para el desarrollo del proyecto será:

1. Fase 1

1.1 Temporización

Semana 1 a semana 4

14 horas de dedicación semanales aproximadamente

1.2 Tareas y actividades

- Planificación inicial
- Diseño del modelo entidad relación

- Diseño arquitectónico
- Desarrollo módulo de registro de datos básicos
- Reunión con los usuarios y pruebas

2. Fase 2

2.1 Temporización

Semana 5 a semana 8

14 horas de dedicación semanales aproximadamente

2.2 Tareas y actividades

- Planificación inicial
- Formación académica: Modulo para registrar uno a uno los títulos obtenidos o en curso
- Reunión con los usuarios y pruebas

3. Fase 3

3.1 Temporización

Semana 9 a semana 12

14 horas de dedicación semanales aproximadamente

3.2 Tareas y actividades

- Planificación inicial
- Experiencia: modulo para registrar una a una la experiencia profesional en el sector público o privado
- Publicaciones: Modulo para registrar la participación en publicaciones, artículos, boletines, etc.
- Reunión con los usuarios y prueba

4. Fase 4

4.1 Temporización

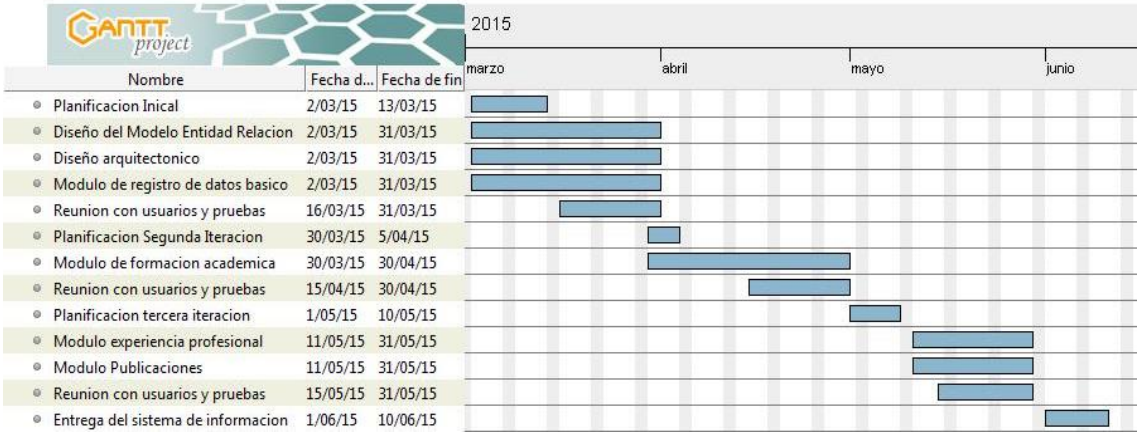
Semana 13

14 horas de dedicación semanales aproximadamente

4.2 Tareas y actividades

- Entrega del sistema de información
- Reunión con los usuarios y prueba final

2.3 Diagrama de Gantt



Nombre	Fecha d...	Fecha de fin
Planificacion Inical	2/03/15	13/03/15
Diseño del Modelo Entidad Relacion	2/03/15	31/03/15
Diseño arquitectonico	2/03/15	31/03/15
Modulo de registro de datos basico	2/03/15	31/03/15
Reunion con usuarios y pruebas	16/03/15	31/03/15
Planificacion Segunda Iteracion	30/03/15	5/04/15
Modulo de formacion academica	30/03/15	30/04/15
Reunion con usuarios y pruebas	15/04/15	30/04/15
Planificacion tercera iteracion	1/05/15	10/05/15
Modulo experiencia profesional	11/05/15	31/05/15
Modulo Publicaciones	11/05/15	31/05/15
Reunion con usuarios y pruebas	15/05/15	31/05/15
Entrega del sistema de informacion	1/06/15	10/06/15

3. ESTADO DEL ARTE

Para la realización de este proyecto se investigo acerca de proyectos ya realizados e implementados y se encontró un sistema de información llamado Sigep que es un sistema de Información y Gestión del Empleo Público al servicio de la administración pública y de los ciudadanos. El sistema contiene información sobre el talento humano al servicio de las organizaciones públicas, en cuanto a datos de las hojas de vida, declaración de bienes, rentas y sobre los procesos propios de las áreas encargadas de administrar al personal vinculado a éstas.

También se encontró una memoria llamada “implementación de una base de datos para el almacenamiento de las hojas de vida de los funcionarios y los contratos realizados por la cámara de comercio de Dosquebradas” esta realizada por Catalina Martínez Saldarriaga de la Universidad Católica Popular del Risaralda

Es un desarrollo propuesto por el departamento de Sistemas de la entidad para acceder más fácilmente a la información contenida en estos documentos. Se realiza con el fin de asegurar la calidad y seguridad de la información guardada en el archivo, de manera que solo puedan acceder a esta las personas autorizadas para ello, y al mismo tiempo se mantenga protegida contra los daños físicos que pueda sufrir en el transcurso del tiempo.

En la ingeniería de software se denomina aplicación web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Este trabajo está enfocado al desarrollo de un software para el manejo del banco de hojas de vida del talento humano académico. La tecnología que se utilizó para desarrollar la aplicación es: Netbeans como IDE y Java + Posgresql.

3.1 Java

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes.

Java es una tecnología que se usa para el desarrollo de aplicaciones que convierten a la Web en un elemento más interesante y útil. Java no es lo mismo que javascript, que se trata de una tecnología sencilla que se usa para crear páginas web y solamente se ejecuta en el explorador.

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés comoWORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

3.2 Netbeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de *código abierto* de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los

proyectos (Actualmente Sun Microsystems es administrado por Oracle Corporation).

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

3.3 Postgresql

Es un Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD.

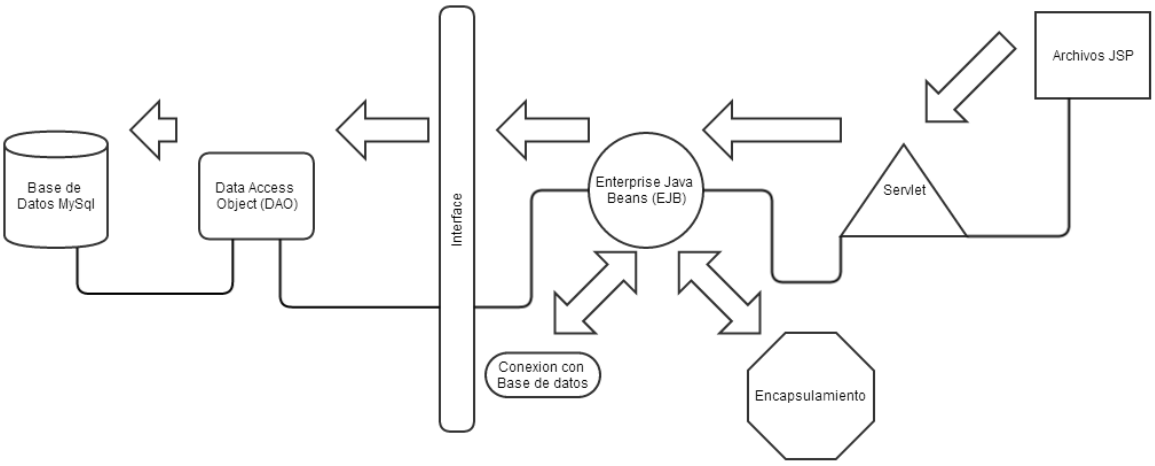
Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

PostgreSQL es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

3.4 Modelo De Desarrollo Dao

Un **Data Access Object** (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una base de datos o un archivo.



3.5 Jsf

Es un estándar de Java hacia la construcción de interfaces de usuario para aplicaciones web que simplifican el desarrollo de aplicaciones web del lado del cliente, JSF está basado en la tecnología Java EE. En el 2009 se dio a conocer la nueva versión JSF 2.0, que contiene algunas características y/o mejoras con respecto a las versiones anteriores (JSF 1.0, JSF 1.1 y JSF 1.2) como son: Mejoras en la navegación: navegación condicional, inspección en tiempo de ejecución en las reglas de navegación. Control de excepciones: permite fácilmente la creación de una página de error que utiliza componentes JSF. Mejoras en la expresión del lenguaje: compatibilidad con métodos arbitrarios incluyendo el paso de parámetros. Validación: es una nueva especificación java desarrollada para la validación de beans. (Burns, 2010)

3.6 Servlets

La tecnología Servlet proporciona las mismas ventajas del lenguaje Java en cuanto a portabilidad (“write once, run anywhere”) y seguridad, ya que un servlet es una clase de Java igual que cualquier otra, y por tanto tiene en ese sentido todas las características del lenguaje. Esto es algo de lo que carecen los programas CGI, ya que hay que compilarlos para el sistema operativo del servidor y no disponen en muchos casos de técnicas de comprobación dinámica de errores en tiempo de ejecución.

Otra de las principales ventajas de los servlets con respecto a los programas CGI, es la del rendimiento, y esto a pesar de que Java no es un lenguaje particularmente rápido. Mientras que los es necesario cargar los programas CGI tantas veces como peticiones de servicio existan por parte de los clientes,

los servlets, una vez que son llamados por primera vez, quedan activos en la memoria del servidor hasta que el programa que controla el servidor los desactiva. De esta manera se minimiza en gran medida el tiempo de respuesta.

Además, los servlets se benefician de la gran capacidad de Java para ejecutar métodos en ordenadores remotos, para conectar con bases de datos, para la seguridad en la información, etc. Se podría decir que las clases estándar de Java ofrecen resueltos muchos problemas que con otros lenguajes tiene que resolver el programador. (2010)

3.7 Jsp

JavaServer Pages (JSP) combinan HTML con fragmentos de Java para producir páginas web dinámicas. Cada página es automáticamente compilada a servlet por el motor de JSP, en primer lugar es recogida y a continuación ejecutada. JSP tiene gran variedad de formas para comunicarse con las clases de Java, servlets, applets y el servidor web; por esto se puede aplicar una funcionalidad a nuestra web a base de componentes. Resumen de la arquitectura de una página JSP Una página JSP es archivo de texto simple que consiste en contenido HTML o XML con elementos JSP. Cuando un cliente pide una página JSP del sitio web y no se ha ejecutado antes, la página es inicialmente pasada al motor de JSP, el cual compila la página convirtiéndola en Servlet, la ejecuta y devuelve el contenido de los resultados al cliente. Es posible ver el código del servlet generado, este código debe estar en el directorio que se informa en la estructura de directorios del servidor. Si nos fijamos en este archivo podemos encontrar las siguientes clases: · JSPPage · HttpJspPage Ellas definen la interface para el compilador de páginas JSP. Nos encontramos también tres métodos:

·JspInit ()

·JspDestroy ()

·_jspService(HttpServletRequest request, HttpServletResponse response)

Los dos primeros métodos pueden ser definidos por el autor de la página JSP, pero el tercer método es una versión compilada de la página JSP, y su creación es responsabilidad del motor de JSP. (García).

4. MARCO TEÓRICO

4.1 Metodología agiles

Para un buen desarrollo de software existen dos tipos de metodologías que inciden en el proceso del desarrollo. Por un lado están las metodologías tradicionales se basan especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas y las herramientas y notaciones que se usaran. Por otra lado existen las metodología agiles los cuales le dan mayor valor a individuo, a la colaboración con el cliente y al desarrollo incremental del software a través de iteraciones muy cortas. A mostrado gran aceptación de las diferentes personas que desarrollan y en proyectos donde los requisitos son muy cambiantes, existen diferente metodologías agiles los cuales son:

4.1.1 Scrum: Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

4.1.2 Crystal Methodologies: Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos, han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una

clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

4.1.3 Dynamic Systems Development Method (DSDM): Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.

4.1.4 Adaptive Software Development (ASD): Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

4.1.5 Feature-Driven Development (FDD): Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.

4.1.6 Lean Development (LD): Definida por Bob Charette's a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios.

4.1.7 eXtreme Programminig (XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

5 IMPLEMENTACIÓN Y DESARROLLO

En el proyecto utilizamos la metodología XP por lo tanto la estudiaremos más detalladamente a continuación:

Dentro de esta metodología cabe resaltar una serie de valores y principios que se debe tener en cuenta y practicarlos durante el tiempo de desarrollo que dure el proyecto.

5.1 Valores

Se considera una disciplina lo cual esta sostenida por valores y principios propios de las metodologías agiles. Existen cuatro valores que cumplen para el trabajo de metodologías livianas:

- La comunicación: es muy importante que dentro del equipo de desarrollo y con el cliente exista un ambiente de colaboración y comunicación. Dentro de esta metodología es tan importante el cliente que hace parte del equipo de desarrollo.
- La simplicidad: consiste en realizar diseños muy sencillos donde lo más relevante sea la funcionalidad necesaria que requiera el cliente, solo se desarrolla lo que el cliente pide y de la forma más sencilla.
- La retroalimentación: es muy importante ya que ayuda a encaminarlo y a darle forma, se presenta de las dos maneras de parte del equipo hacia el cliente con el fin de brindarle información acerca de la evolución del sistema y viceversa en los aporte a la construcción del proyecto.
- El coraje: sirve para estar preparado s los diferentes cambios que se presentaran en el transcurso de la actividad, cada integrante debe tener el

valor de dar a conocer las dudas o posibles problemas que se encuentre en la realización del proyecto.

5.2 Practicas

Dentro de esta metodología se plantean doce reglas, una de sus principales características es su simplicidad y su enfoque en la practicidad, además cada regla es el complemento de la anterior:

- ✓ El juego de la planificación: Hay una comunicación frecuente el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- ✓ Entregas pequeñas: Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses.
- ✓ Metáfora: El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).
- ✓ Diseño simple: Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
- ✓ Pruebas: La producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.
- ✓ Refactorización (Refactoring): Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.
- ✓ Programación en parejas: Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores).

- ✓ Propiedad colectiva del código: Cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- ✓ Integración continua: Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- ✓ 40 horas por semana: Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.
- ✓ Cliente in-situ: El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.
- ✓ Estándares de programación: XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

El mérito de XP es integrar las doce reglas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

5.3 Análisis De Requisitos

La universidad de Cundinamarca tiene la necesidad de contar con un sistema que permita gestionar el banco de hojas de vida de sus docentes, y que a su vez sea fácil de manejar. La finalidad del software es que el docente pueda llevar a cabo un registro de todos los datos de su hoja de vida.

Para dicho registro se necesitaran datos como los siguientes:

5.3.1 Básicos: Para poder dar ingresar un docente se tendrán en cuenta: una clave o número de cedula del docente, el nombre, el apellido, genero, nacionalidad, libreta militar, fecha y lugar de nacimiento, país, departamento, municipio, dirección de correspondencia, email, teléfono, estado civil.

5.3.2 Académicos: Se deben ingresar los datos académicos de los docentes. Para este módulo se asignaran datos como: la

modalidad académica (Doctorado, maestría, especialización, pregrado), año de terminación, numero de la tarjeta profesional, idiomas.

5.3.3 Actividades de perfeccionamiento: Se debe registrar otros tipos de estudio o capacitaciones realizadas como diplomados, cursos, seminarios, certificaciones. Se requieren los siguientes datos: Entidad donde se realizó, nombres, fechas, intensidad horaria.

5.3.4 Experiencia Calificada: En este módulo se debe registrar la experiencia profesional en áreas como: profesional diferente a la docencia, docencia universitaria, investigación, dirección académica. Se requiere para este módulo datos como: Entidad (pública o privada), dirección entidad, teléfono, cargo, correo electrónico, fecha de ingreso, fecha de retiro.

5.3.5 Productividad Académica: Se debe registrar en este módulo los diferentes tipos de producción académica: Artículos, comunicación corta, informes de caso, revisiones de tema, cartas al editor o editoriales, videos, fotografía, libros, traducciones, premios, patentes, obras artísticas, producción técnica, producción de software, producción de videos, cinematográfica o fonográfica, obras artísticas, ponencias, publicaciones, artículos revistas no indexadas, estudios postdoctorales, traducciones publicadas, direcciones de tesis.

A continuación esta información será cargada a la base de datos del sistema de hojas de vida y a través de este sistema serán evaluadas las personas que continuarán en los procesos de selección de docentes para la universidad.

5.4 Historias de Usuario

Dada la metodología para desarrollo de software las siguientes son historias de usuario, las cuales están en mejoramiento.

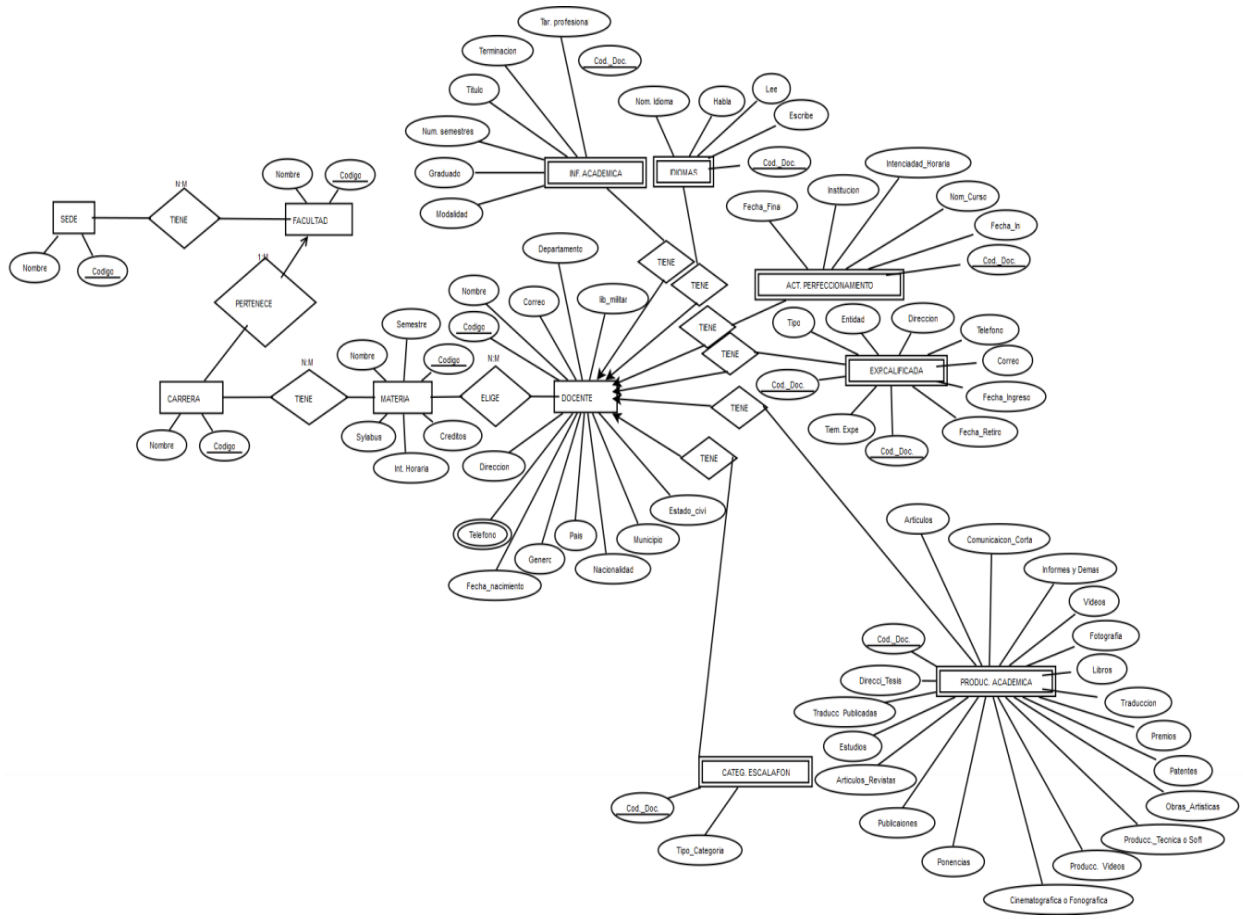
Historia de Usuario	
Número: 1	**Usuario:** Docente
Nombre historia: registro hoja de vida	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** **N/A**
Puntos estimados: N/A	**Iteración asignada: N/A**
Programador responsable: N/A	
Descripción: Ingresar a la aplicación, donde tendrá un menú para escoger la sede, la facultad, la carrera y la materia a la cual quiere registrar todos sus datos personales y experiencia laboral, según el programa académico seleccionado. Después de completado el formulario, el usuario dará clic en finalizar y el formulario quedara automáticamente guardado en la base de datos.	
Criterios de aceptación: - El docente podrá escoger el programa en el cual desea registrar la hoja de vida	

Historia de Usuario	
Número: 2	**Usuario:** Docente
Nombre historia: interfaz grafica	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** **N/A**
Puntos estimados: N/A	**Iteración asignada: N/A**
Programador responsable: N/A	
Descripción: Crear una interfaz gráfica donde el usuario pueda navegar y seleccionar la sede, facultad, carrera y materia en un entorno gráfico.	
Criterios de aceptación: - El docente podrá escoger la sede, facultad, carrera y materia a través de la interfaz gráfica.	

5.5 Diseño De Solución

5.5.1 Diseño De Base De Datos

5.5.1.1 Modelo entidad relación



5.5.1.2 Esquema Relacional

```
CREATE TABLE act_perfeccionamiento (
   codigo_doc_act character varying(30) NOT NULL,
   institucion character varying(50),
   nombre_curso character varying(50),
   intensidad_horaria character varying(30),
   fecha_ini date,
   fecha_fin date,
   codigo_docente character varying(30) NOT NULL
);
CREATE TABLE carrera (
   codigo_carrera character varying(30) NOT NULL,
   nombre_carrera character varying(40),
   codigo_facultad character varying(30) NOT NULL
);
CREATE TABLE carrera_materia (
   codigo_carrera character varying(30) NOT NULL,
   codigo_materia character varying(30) NOT NULL
);
CREATE TABLE categ_escalafon (
   codigo_doc_cat character varying(30) NOT NULL,
   tipo_categoria character varying(30),
   codigo_docente character varying(30) NOT NULL
);
CREATE TABLE docente (
```

```
    codigo_docente character varying(30) NOT NULL,
    primer_nombre character varying(30),
    segundo_nombre character varying(30),
    primer_apellido character varying(30),
    segundo_apellido character varying(30),
    telefono character varying(30),
    correo character varying(40),
    pais character varying(30),
    departamento character varying(30),
    municipio character varying(30),
    libreta_militar character varying(30),
    fecha_nac date,
    genero character varying(10),
    estado_civil character varying(30)
);
CREATE TABLE exp_calificada (
    codigo_doc_exp character varying(30) NOT NULL,
    tipo character varying(30),
    entidad character varying(50),
    direccion character varying(40),
    telefono character varying(30),
    correo character varying(40),
    fecha_ingreso date,
    fecha_retiro date,
    tiempo_experiencia character varying(30),
    codigo_docente character varying(30) NOT NULL
);
TABLE facultad (
    codigo_facultad character varying(30) NOT NULL,
    nombre_facultad character varying(40)
);
CREATE TABLE idioma (
    codigo_doc_idi character varying(30) NOT NULL,
    nombre_idioma character varying(40),
    habla_idioma character varying(30),
    lee_idioma character varying(30),
    escribe_idioma character varying(30),
    codigo_docente character varying(30) NOT NULL
);
CREATE TABLE informacion_academica (
    codigo_doc_inf character varying(30) NOT NULL,
    tarjeta_profesional character varying(30),
    terminacion character varying(30),
    titulo character varying(100),
    numero_semestres character varying(10),
    graduado character varying(10),
    modalidad character varying(30),
    codigo_docente character varying(30) NOT NULL
);
CREATE TABLE materia (
    codigo_materia character varying(30) NOT NULL,
    nombre_materia character varying(40),
    semestre character varying(10),
    sylabus character varying(100),
```

```
    creditos character varying(10),
    intensidad_hor character varying(30)
);
CREATE TABLE materia_docente (
    codigo_materia character varying(30) NOT NULL,
    codigo_docente character varying(30) NOT NULL
);
CREATE TABLE produc_academica (
    codigo_doc_produc character varying(30) NOT NULL,
    articulo character varying(30),
    comunicacion_corta character varying(30),
    informes character varying(30),
    videos character varying(30),
    fotografia character varying(30),
    libros character varying(30),
    traduccion character varying(30),
    premios character varying(30),
    patentes character varying(30),
    obras_artisticas character varying(30),
    produccion_tecnica character varying(30),
    produccion_videos character varying(30),
    cinematografica character varying(30),
    ponencias character varying(30),
    publicaciones character varying(30),
    articulos_revisados character varying(30),
    estudios character varying(30),
    traducc_publicas character varying(30),
    direcci_tesis character varying(30),
    codigo_docente character varying(30) NOT NULL
);
CREATE TABLE sede (
    codigo_sede character varying(30) NOT NULL,
    nombre_sede character varying(40)
);
CREATE TABLE sede_facultad (
    codigo_sede character varying(30) NOT NULL,
    codigo_facultad character varying(30) NOT NULL
);
CREATE TABLE usuario (
    cedula_usuario integer NOT NULL,
    nombre_usuario character varying(40),
    contrasenia character varying(30),
    correo character varying(30)
);
ALTER TABLE ONLY act_perfeccionamiento
    ADD CONSTRAINT pk_act_perfeccionamiento PRIMARY KEY (codigo_doc_act);
ALTER TABLE ONLY carrera
    ADD CONSTRAINT pk_carrera PRIMARY KEY (codigo_carrera);
ALTER TABLE ONLY carrera_materia
    ADD CONSTRAINT pk_carrera_materia PRIMARY KEY (codigo_carrera,
codigo_materia);
ALTER TABLE ONLY categ_escalafon
    ADD CONSTRAINT pk_categ_escalafon PRIMARY KEY (codigo_doc_cat);
ALTER TABLE ONLY docente
```

```
   ADD CONSTRAINT pk_docente PRIMARY KEY (codigo_docente);
ADD CONSTRAINT pk_exp_calificada PRIMARY KEY (codigo_doc_exp);
ADD CONSTRAINT pk_facultad PRIMARY KEY (codigo_facultad);
ALTER TABLE ONLY idioma
ALTER TABLE ONLY informacion_academica
   ADD CONSTRAINT pk_informacion_academica PRIMARY KEY (codigo_doc_inf);
ADD CONSTRAINT pk_materia PRIMARY KEY (codigo_materia);
ALTER TABLE ONLY materia_docente
   ADD CONSTRAINT pk_materia_docente PRIMARY KEY (codigo_materia,
codigo_docente);
ADD CONSTRAINT pk_produc_academica PRIMARY KEY (codigo_doc_produc);
ADD CONSTRAINT pk_sede PRIMARY KEY (codigo_sede);
ADD CONSTRAINT pk_sede_facultad PRIMARY KEY (codigo_sede,
ADD CONSTRAINT pk_usuario PRIMARY KEY (cedula_usuario);
ADD CONSTRAINT fk_carrera_materia_codigo_materia FOREIGN KEY
(codigo_materia) REFERENCES materia(codigo_materia);
ADD CONSTRAINT fk_cod_docente_act FOREIGN KEY (codigo_docente)
REFERENCES docente(codigo_docente) ON UPDATE RESTRICT ON DELETE
RESTRICT;
ADD CONSTRAINT fk_cod_docente_cat FOREIGN KEY (codigo_docente)
REFERENCES docente(codigo_docente) ON UPDATE RESTRICT ON DELETE
ADD CONSTRAINT fk_cod_docente_exp FOREIGN KEY (codigo_docente)
REFERENCES docente(codigo_docente) ON UPDATE RESTRICT ON DELETE
RESTRICT;
ADD CONSTRAINT fk_cod_docente_idio FOREIGN KEY (codigo_docente)
REFERENCES docente(codigo_docente) ON UPDATE RESTRICT ON DELETE
RESTRICT;
ADD CONSTRAINT fk_cod_docente_info FOREIGN KEY (codigo_docente)
REFERENCES docente(codigo_docente) ON UPDATE RESTRICT ON DELETE
RESTRICT;
ADD CONSTRAINT fk_cod_docente_produc FOREIGN KEY (codigo_docente)
REFERENCES docente(codigo_docente) ON UPDATE RESTRICT ON DELETE
RESTRICT;
ADD CONSTRAINT fk_materia_docente_codigo_docente FOREIGN KEY
(codigo_docente) REFERENCES docente(codigo_docente);
ADD CONSTRAINT fk_materia_docente_codigo_materia FOREIGN KEY
(codigo_materia) REFERENCES materia(codigo_materia);
ADD CONSTRAINT fk_sede_facultad_codigo_facultad FOREIGN KEY
(codigo_facultad) REFERENCES facultad(codigo_facultad);
ADD CONSTRAINT fk_sede_facultad_codigo_sede FOREIGN KEY (codigo_sede)
REFERENCES sede(codigo_sede);
```

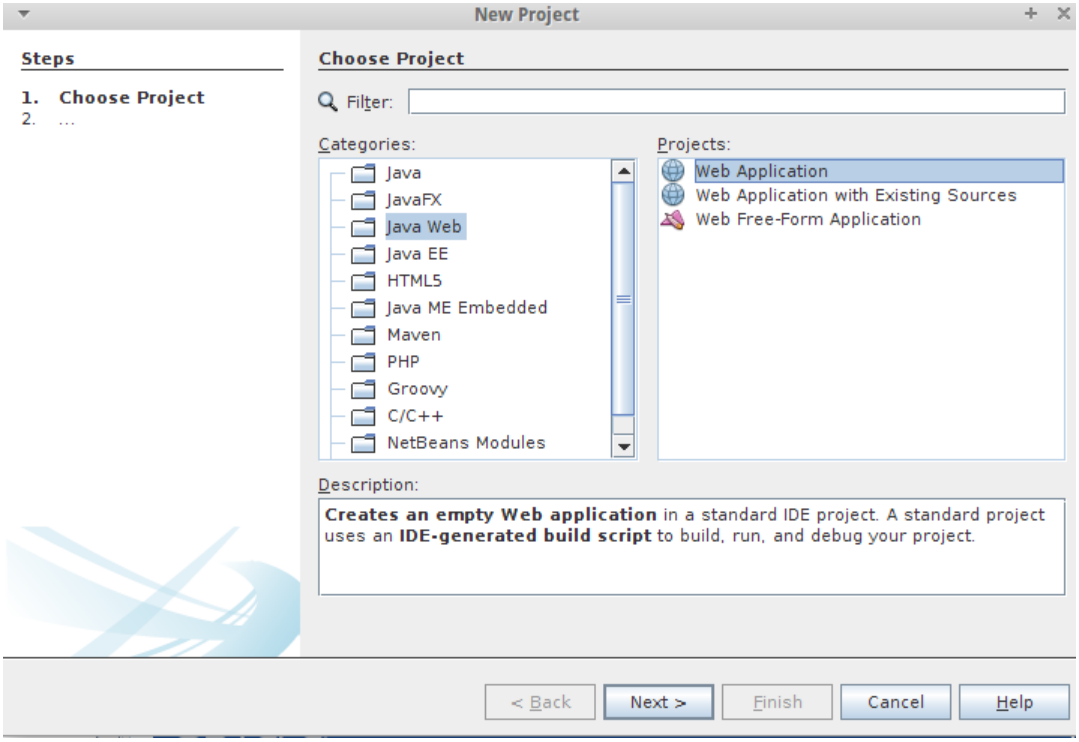
6 RESULTADOS

6.1 Desarrollo del proyecto

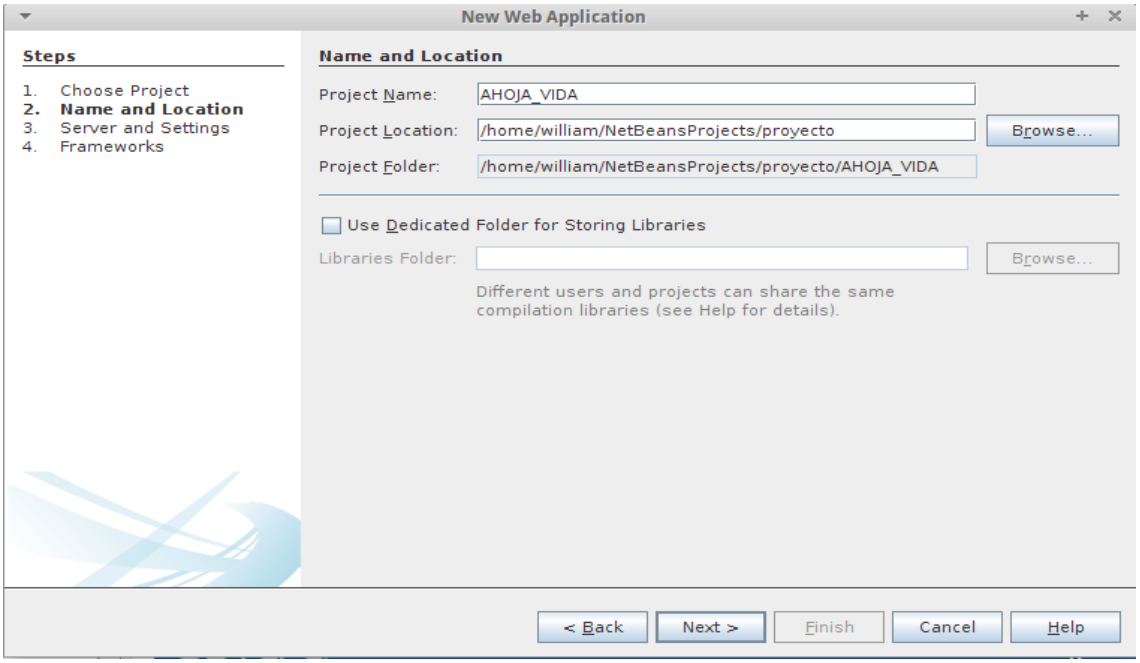
Tecnología J2EE

- JSF
- Primeface
- Hibernate
- JPA
- Patron de diseño DAO

1. Crear el proyecto.
   Seleccionamos la categoría java web, y el proyecto web Application. Y le damos next.

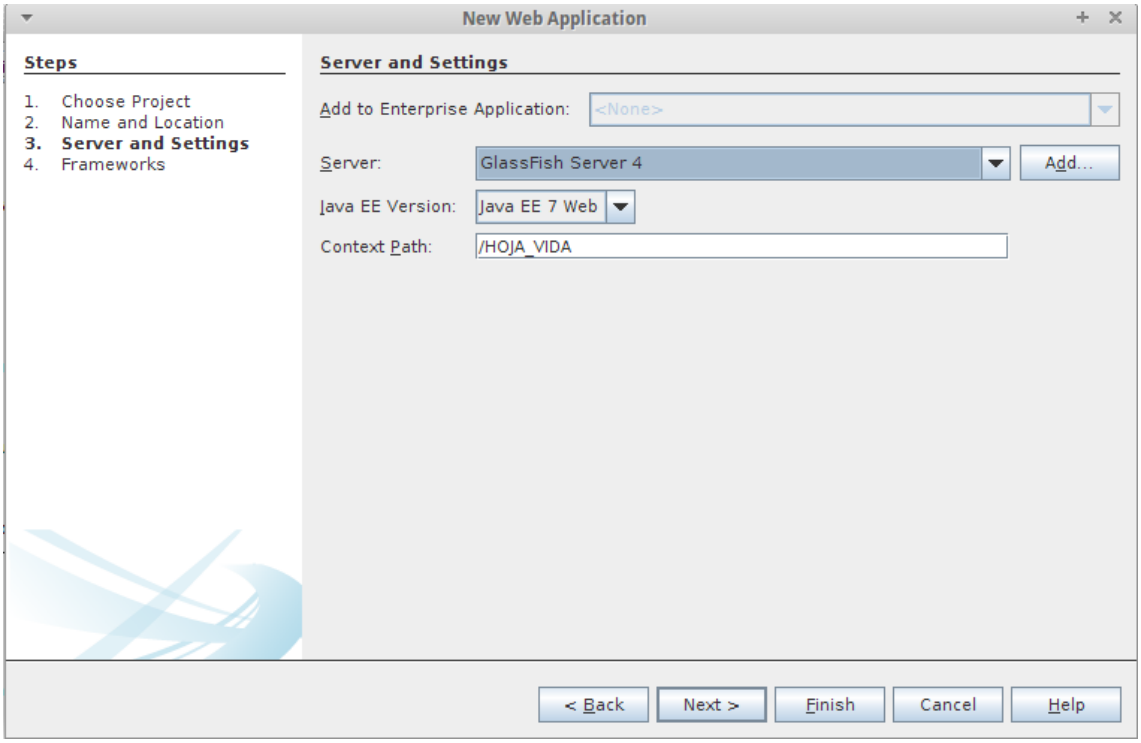


2. Colocar un nombre al proyecto. y next.

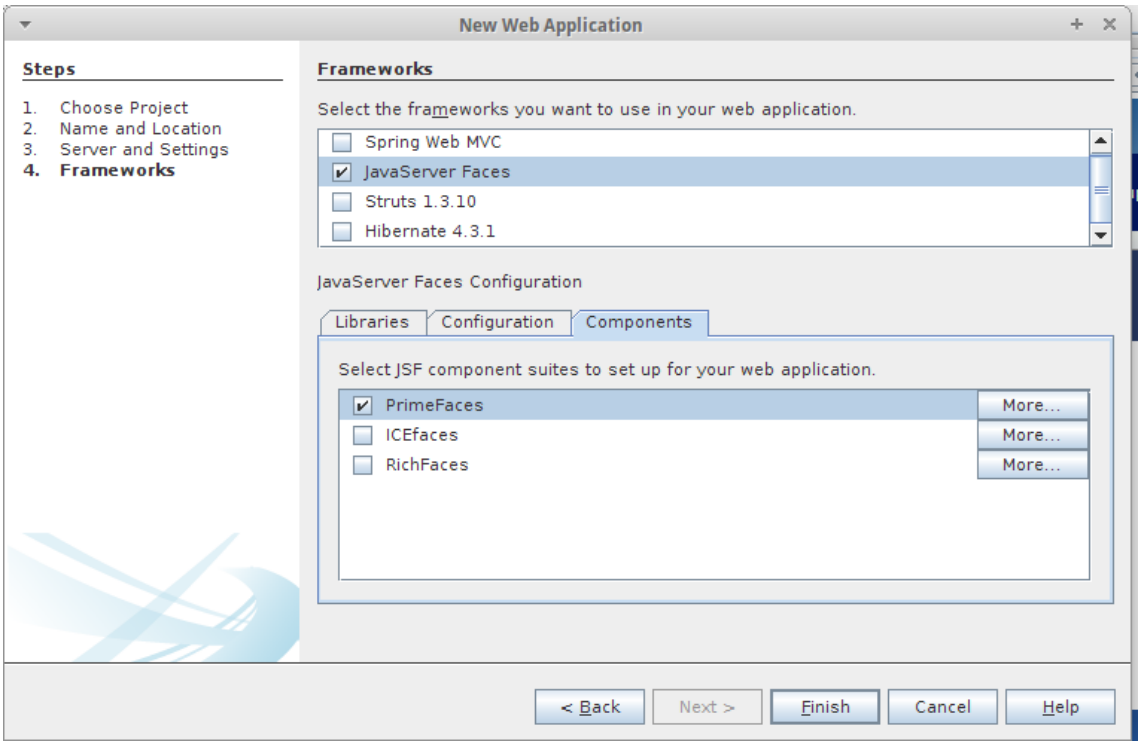


3. Servidor y configuraciones
   - Escogemos servidor : GlassFish Server 4

- Versión java EE: java EE 7 WEB

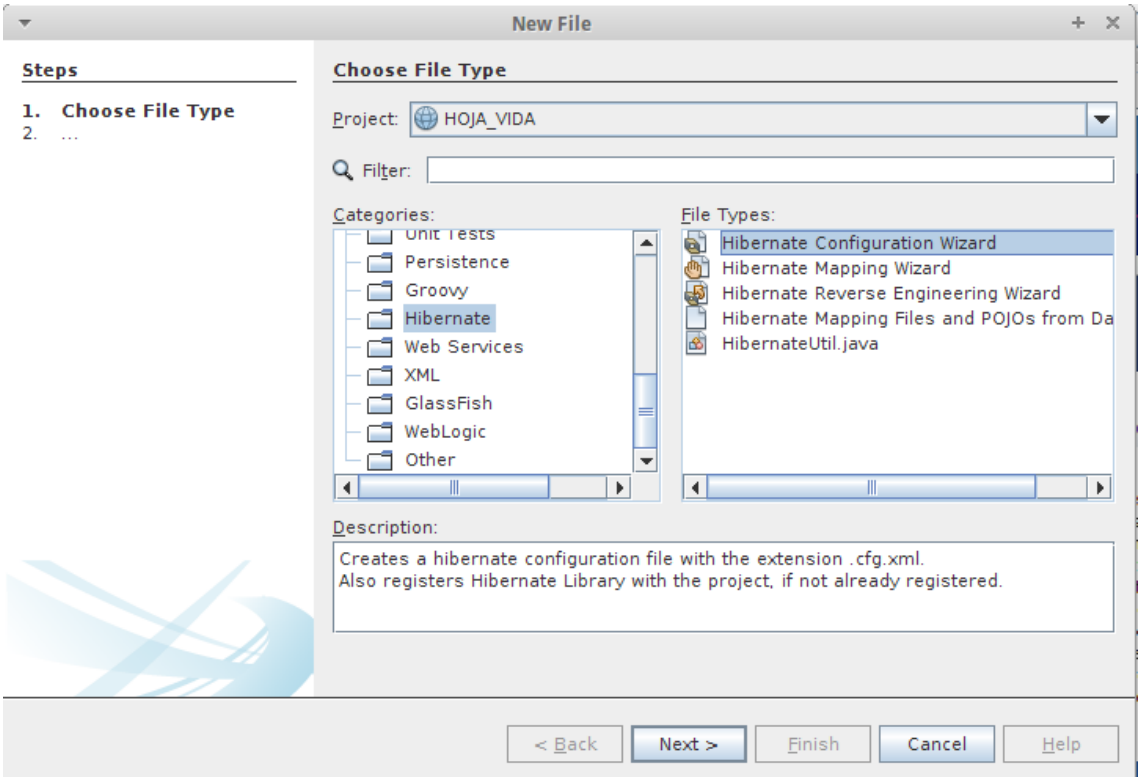


4. En la siguiente ventana seleccionamos la casilla de javaServer Faces como Frameworks y en la parte de abajo en el menú de navegación seleccionamos Components y marcamos la opción de Primeace para poder utilizarlo en nuestra vistas. Y le damos finish

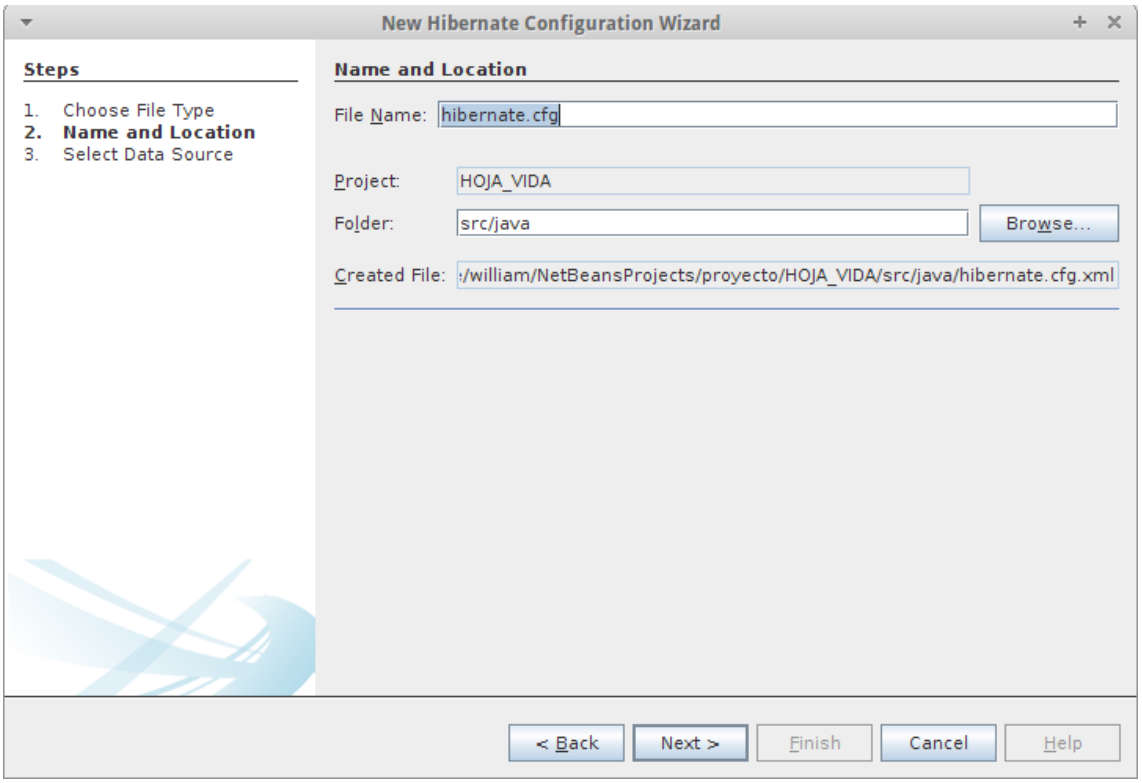


5. El siguiente paso es crear la configuración de Hibernate.
   - En categorías seleccionamos Hibernate

- Y en tipo de archivo seleccionamos Hibernate Configuration Wizard. Y next

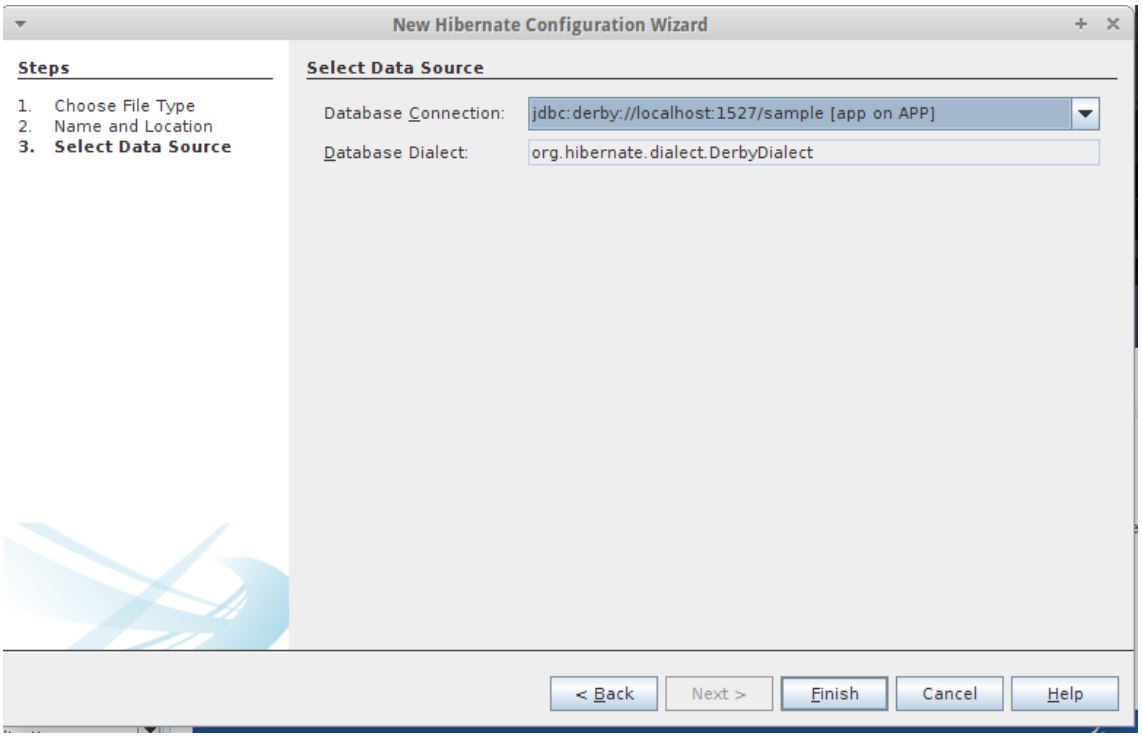


6. Asignamos un nombre al archivo de configuración. Y next

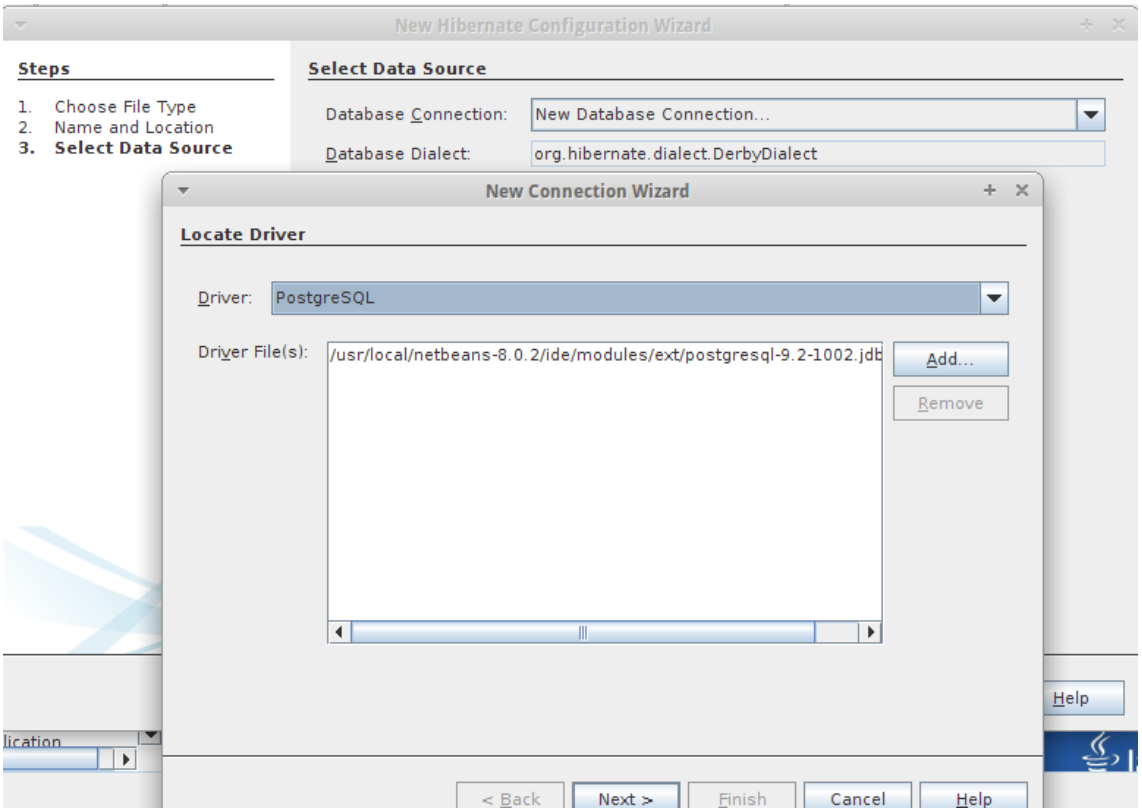


7. Creamos una nueva conexión a la base de datos.

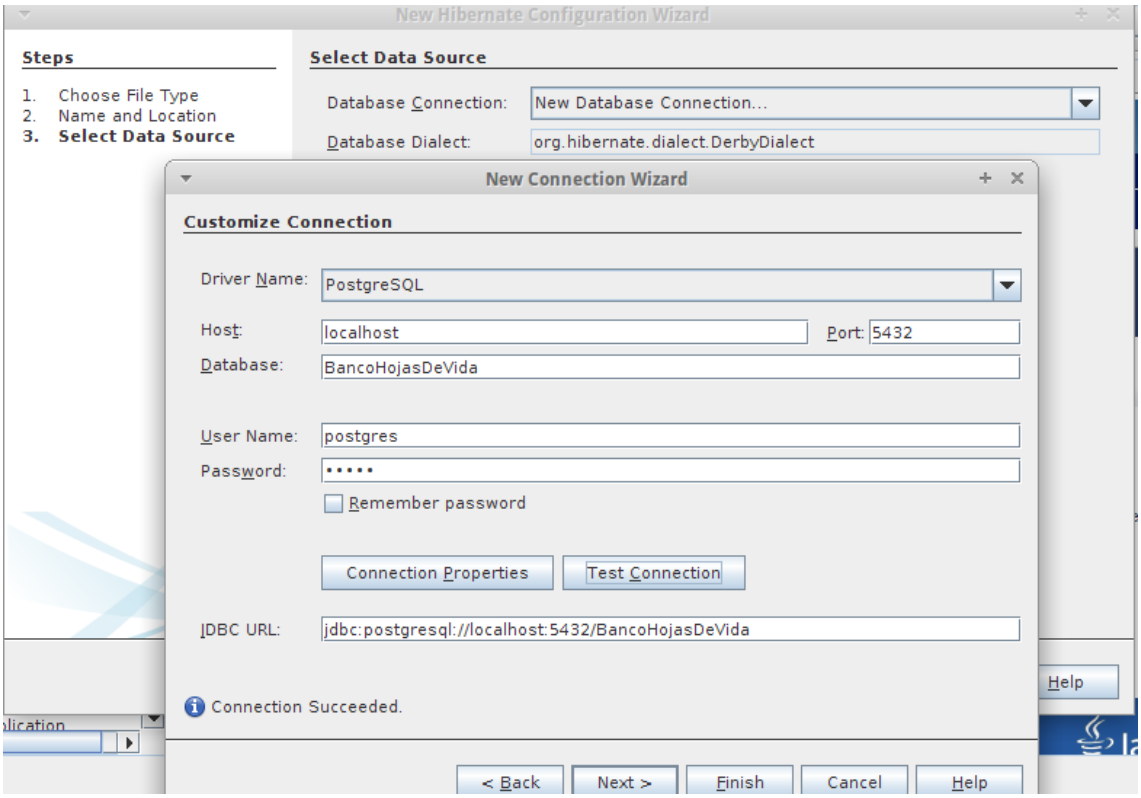
- En database Connection desplegamos y seleccionamos new Database Connection



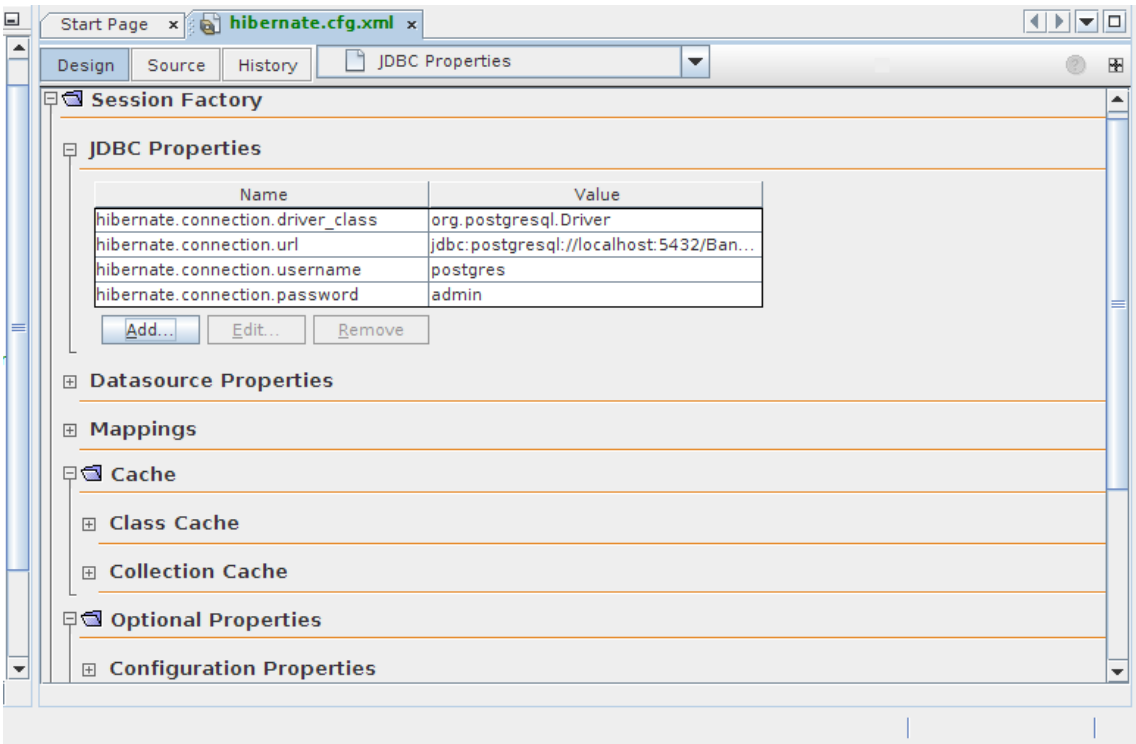
8. Escogemos el driver para poder conectarnos con la base de datos en este caso postgresSQL.



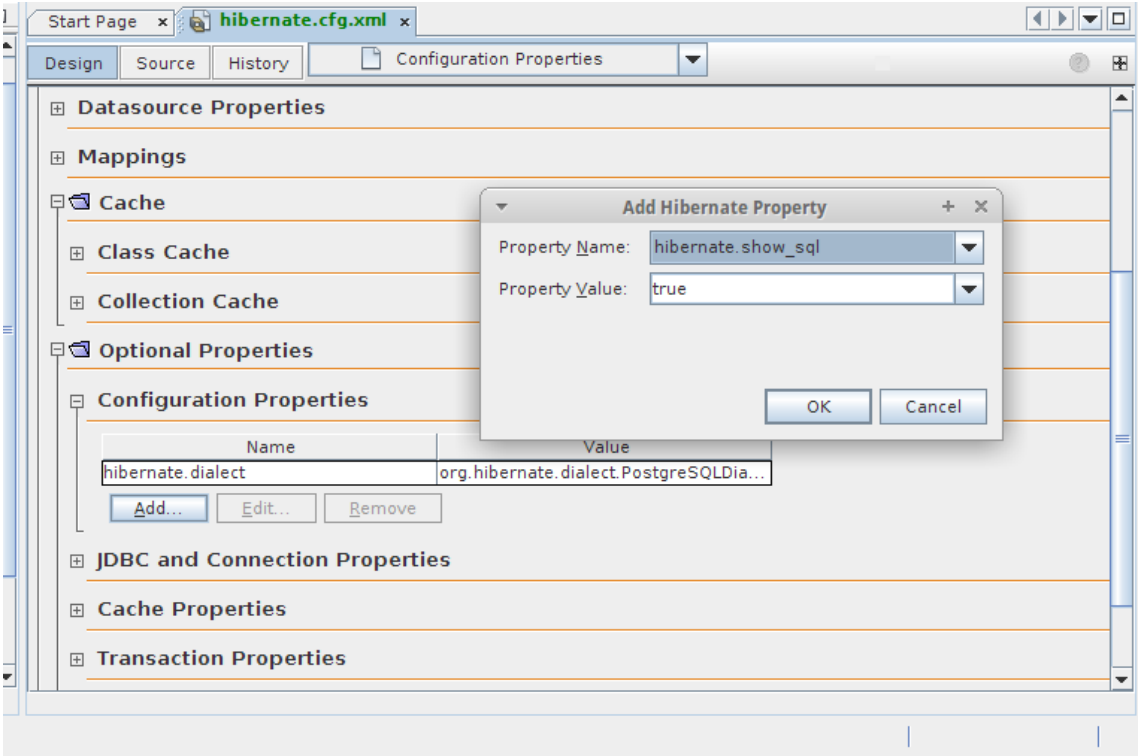
9. Escribimos todos los campos necesarios para conexión a la base de datos. Y por último oprimimos en Test Connection para verificar que la conexión se hizo de forma correcta.



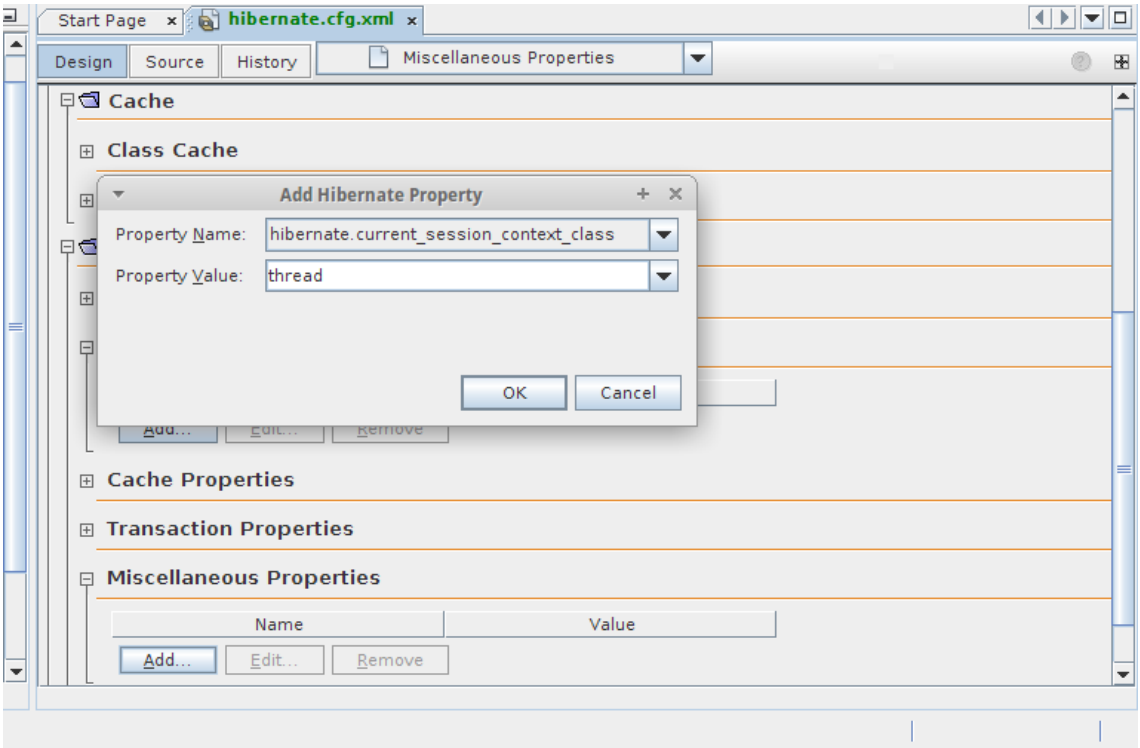
10. Se nos abrirá el archivo Hibernate.cfg.xml donde nos aparece las configuraciones que tiene Hibernate.



11. Vamos a la parte de Optional Properties y en configuration Properties oprimimos en Add. En la ventana que se abre escojemos las opciones de la imagen y oprimimos OK.

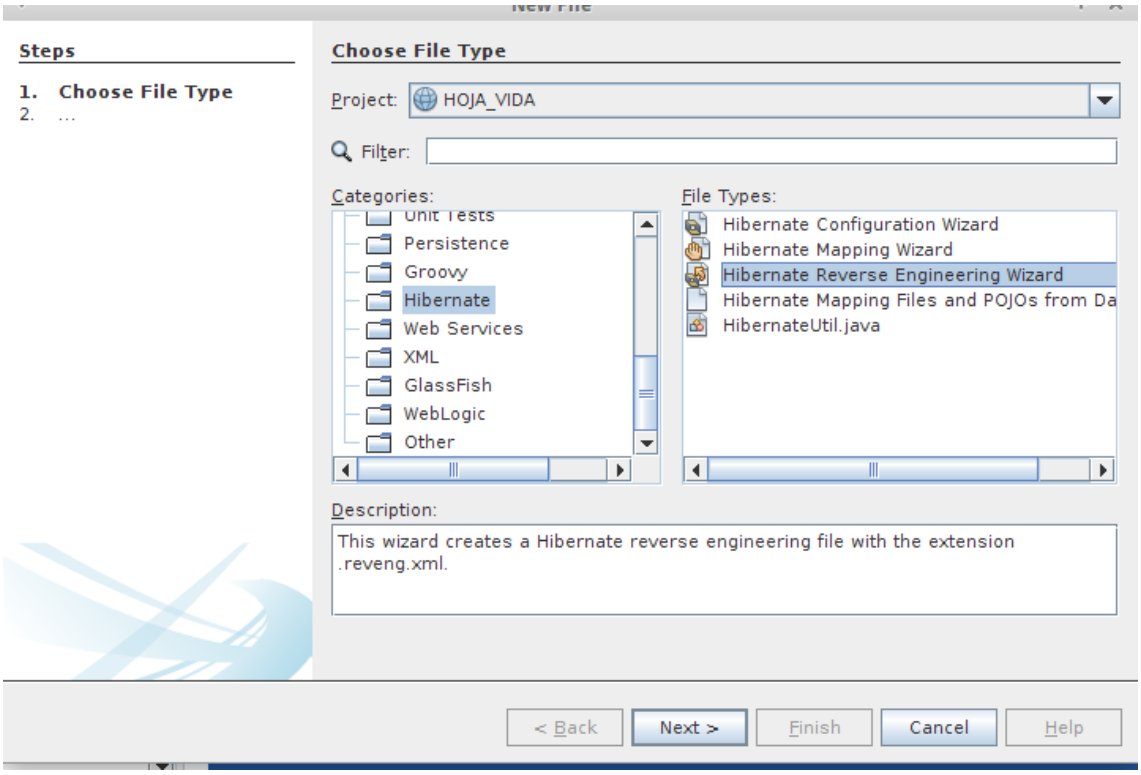


12. Vamos agregar otra propiedad.
    - Seleccionamos miscellaneous Properties, oprimimos en add y en la ventana seleccionamos las propiedades que están en la imagen

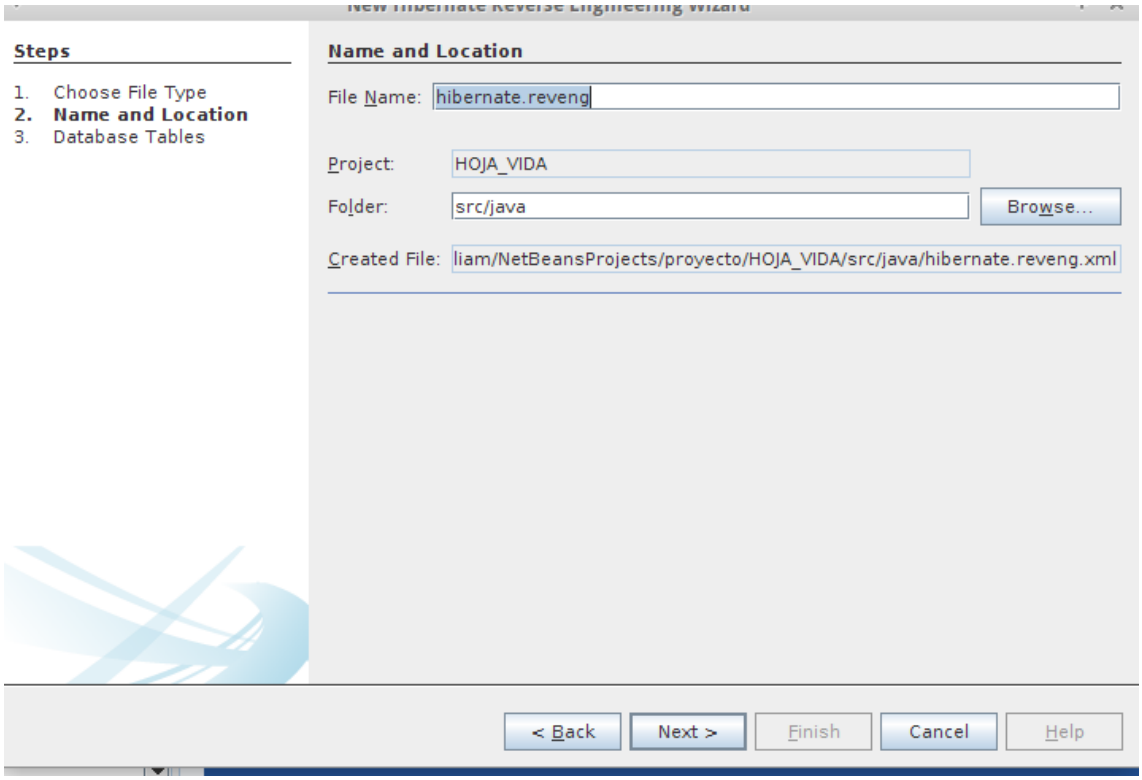


13. El siguiente paso es hacer la ingeniería Inversa.

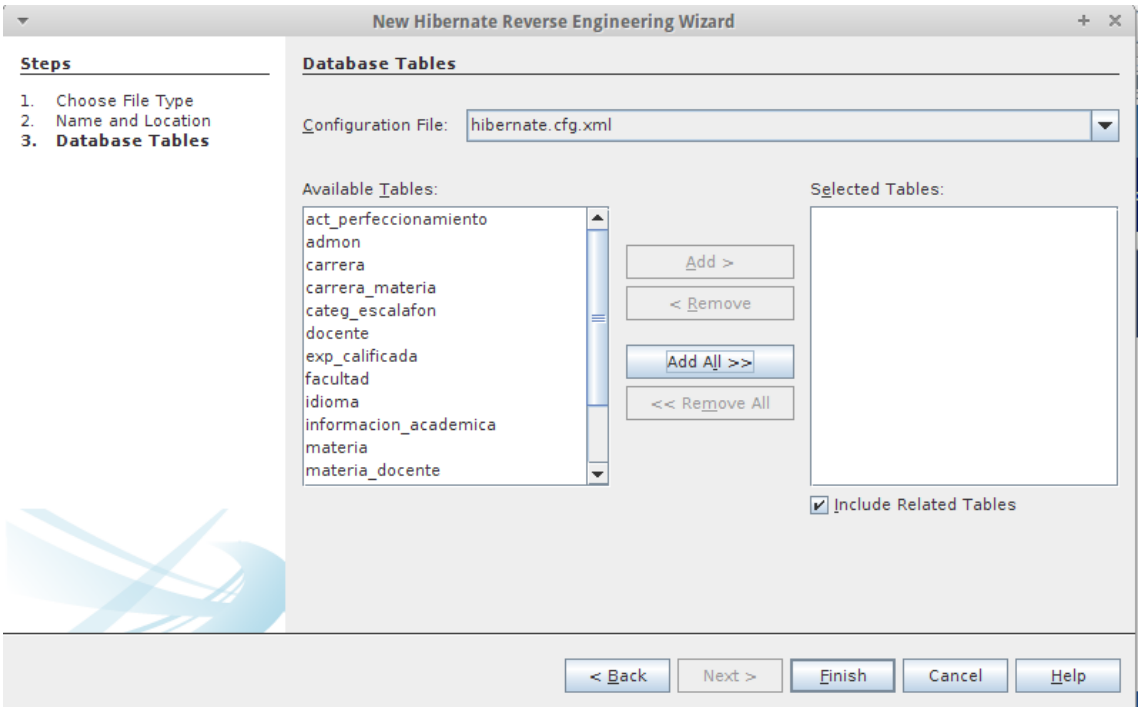
- Creamos un archivo.seleccionamos en categorías Hibernate y en tipo de archivo Hibernate Reverse Engineering Wizard. Y next.



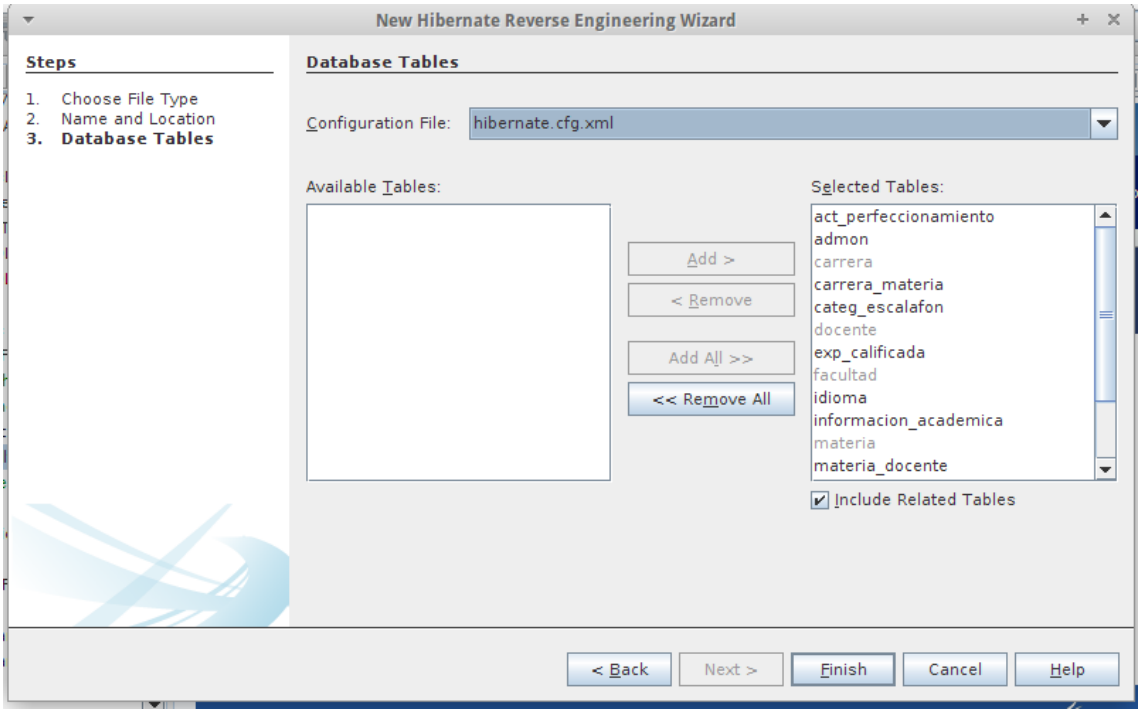
14. asignamos un nombre al archivo. Y oprimimos en next



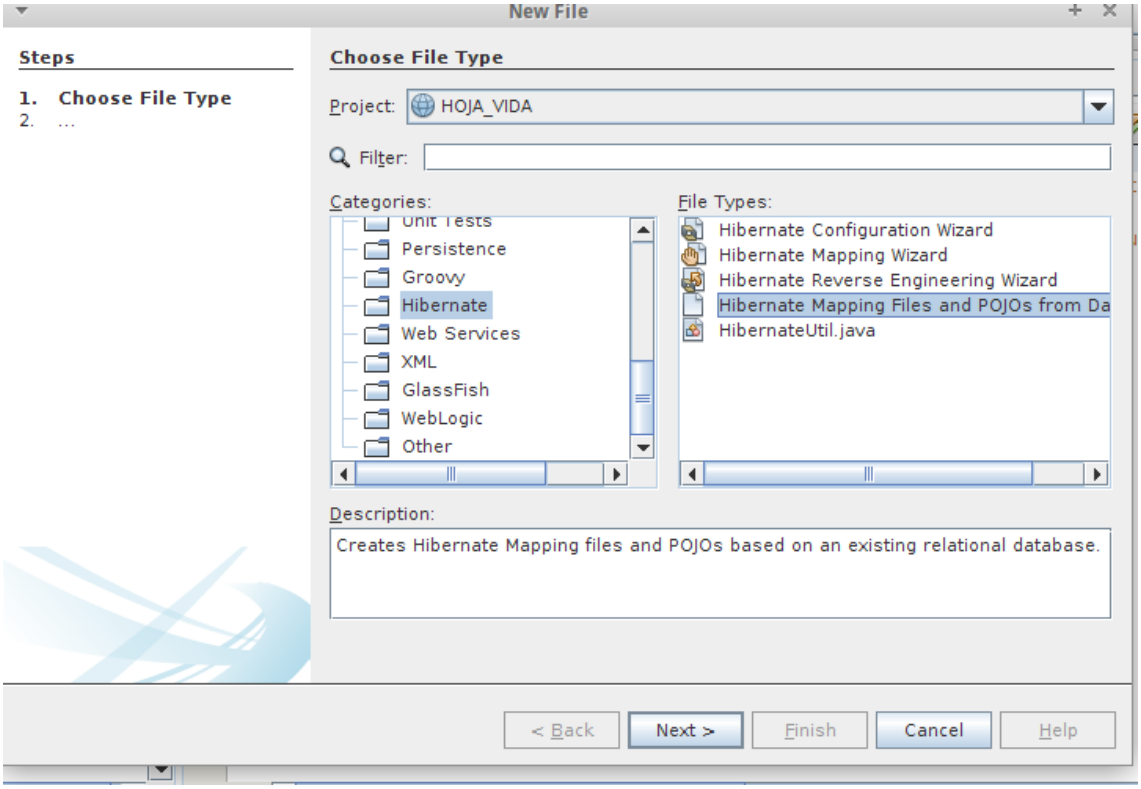
15. nos aparecerá en la parte izquierda las tablas que tenemos creadas en nuestra base de datos Postgres.



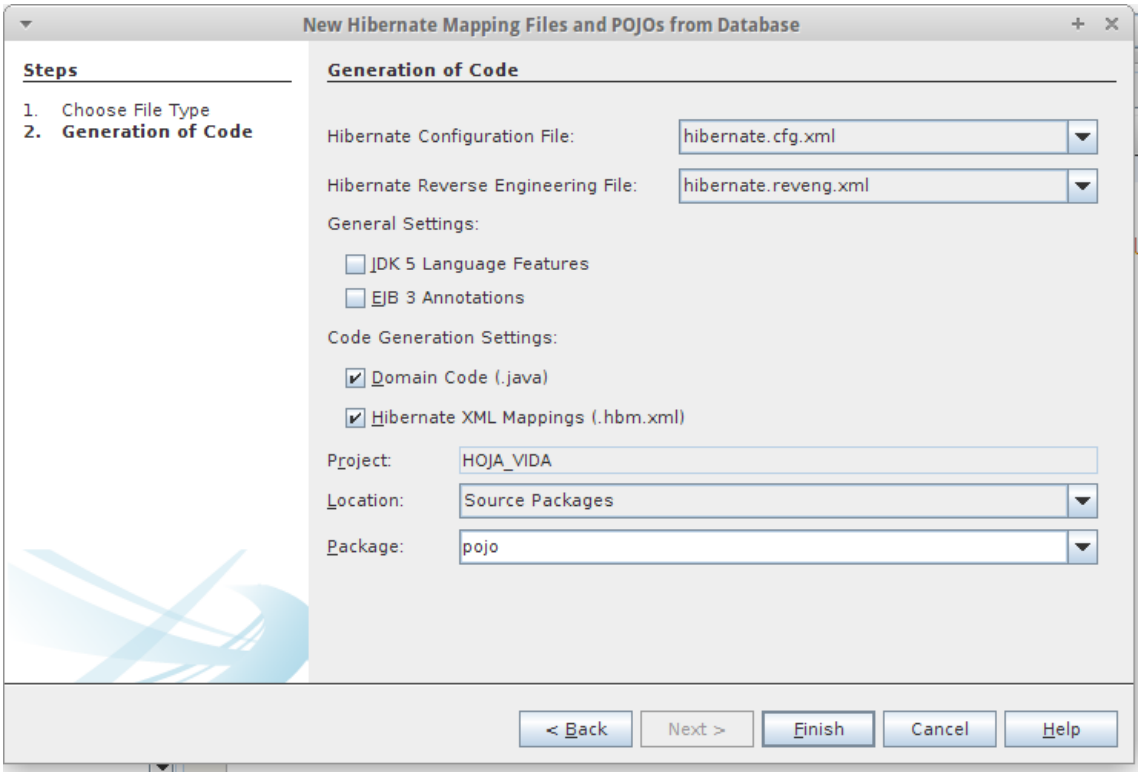
16. seleccionamos todas las tablas: add All. Y oprimimos en finish.
Ya tenemos nuestro archivo de ingeniería inversa.



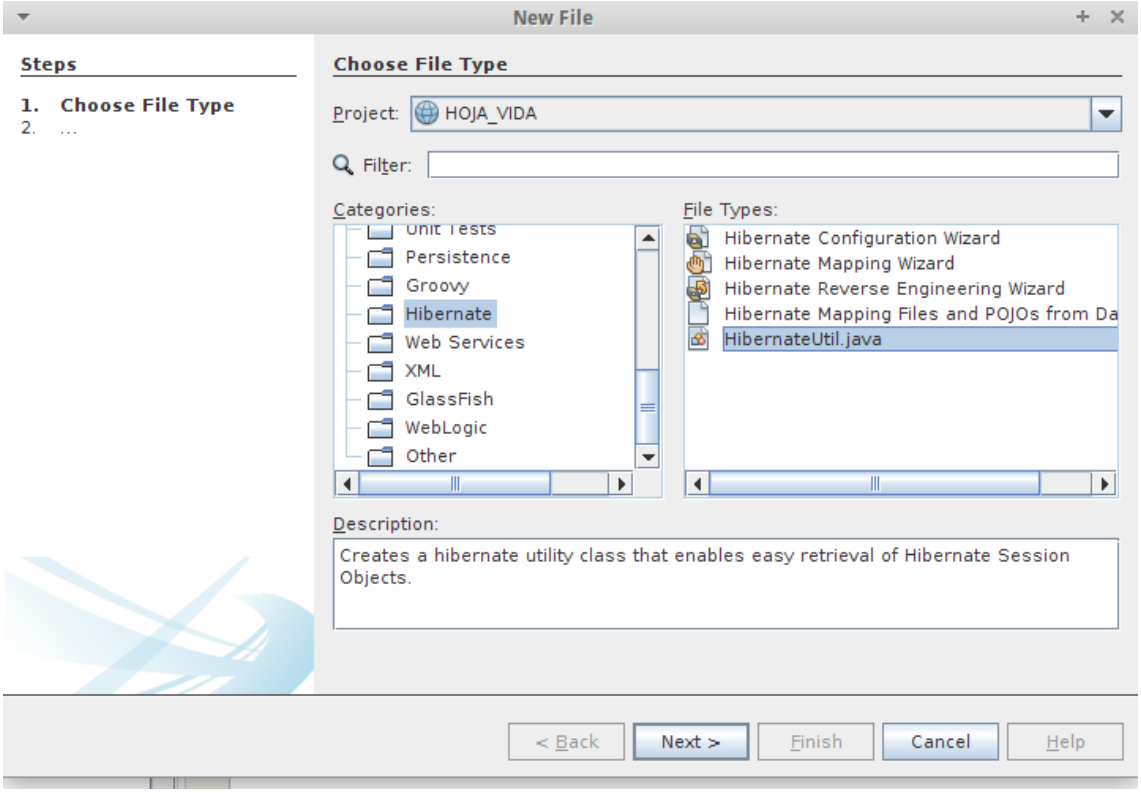
17. el siguiente paso es crear los pojos con hibernate.
    - Seleccionamos en la categoría Hibernate y en tipo de archivo escogemos Hibernate Mapping Files and POJOS.



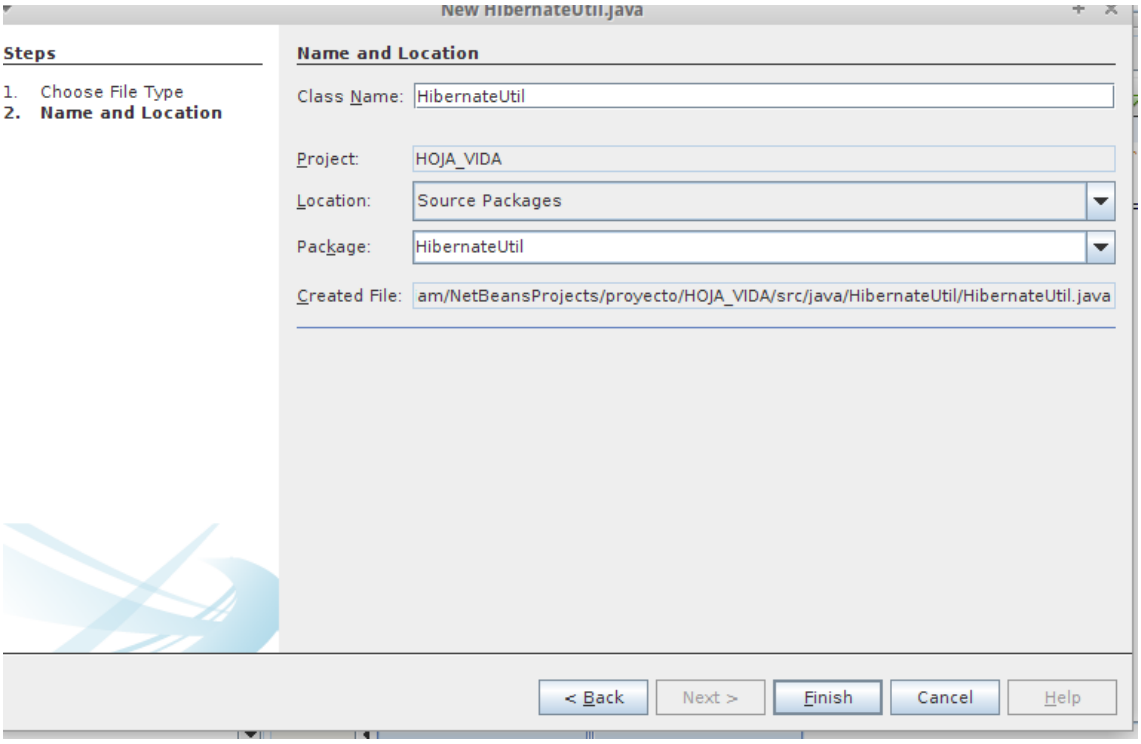
18. En esta ventana creamos la carpeta donde se almacenaran los pojos.



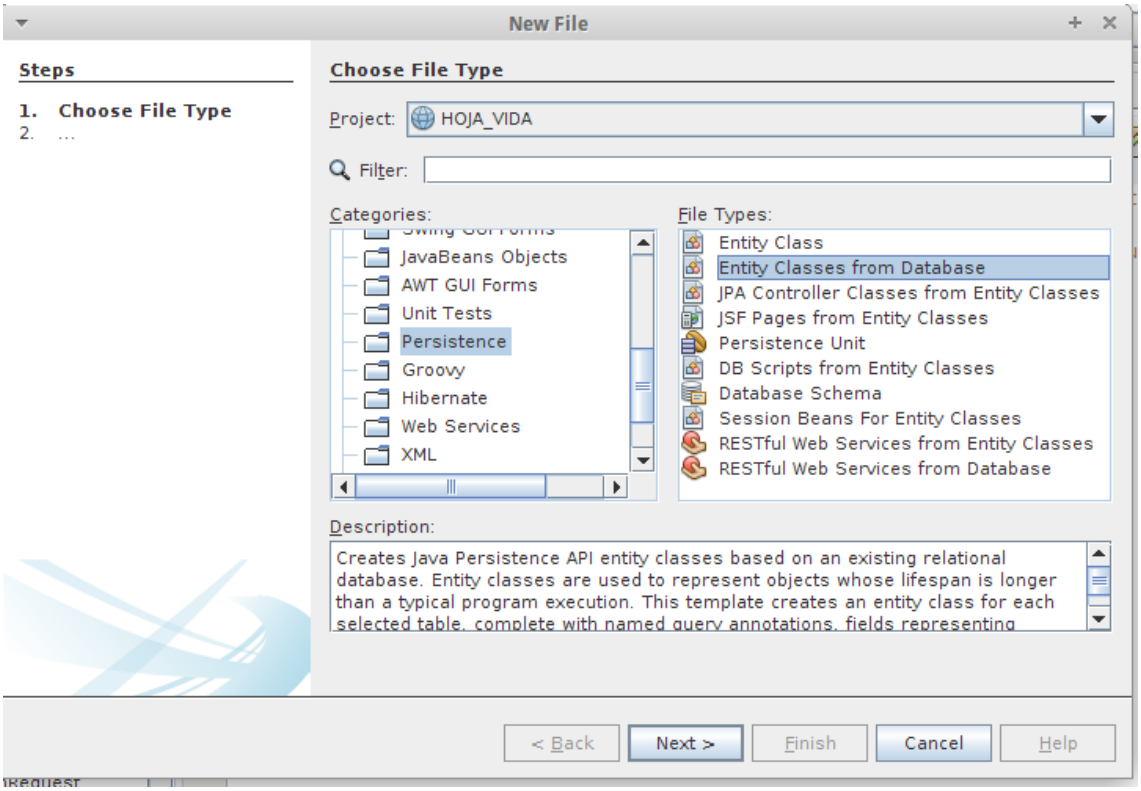
19. Creamos un archivo. Seleccionamos en categorías Hibernate y en tipo de archivo HibernateUtil.java



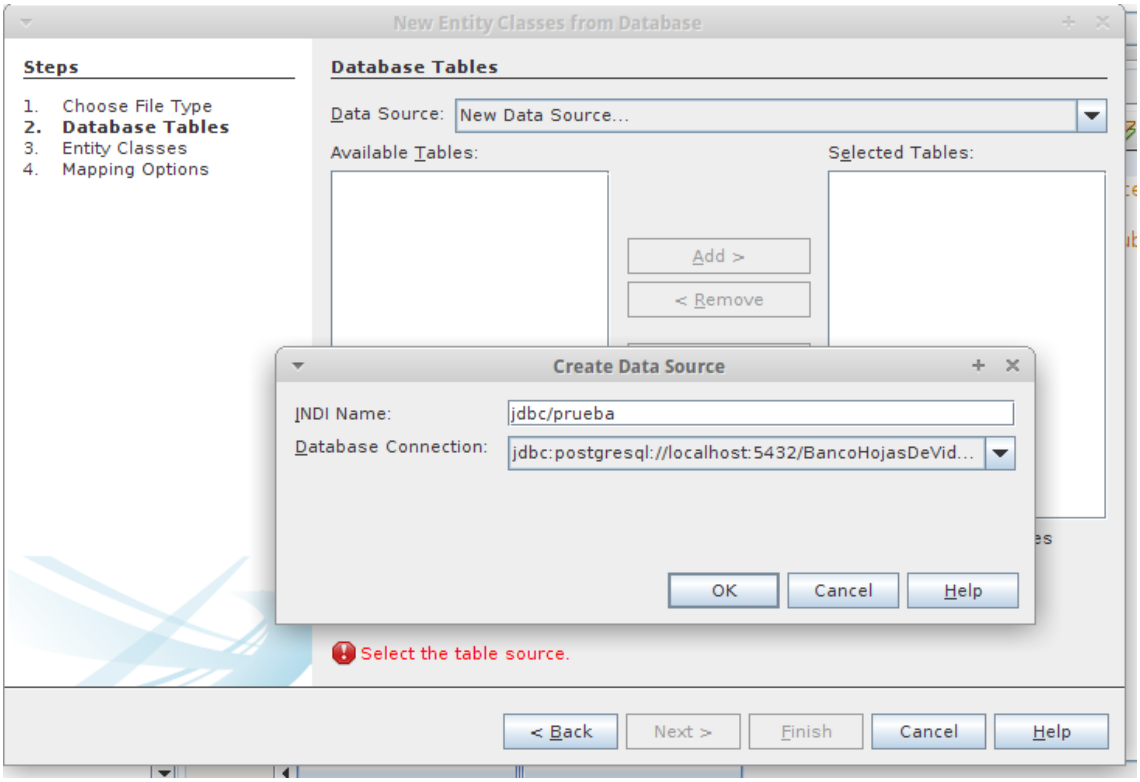
20. Asignamos un nombre y una carpeta al archivo y oprimimos en finish.



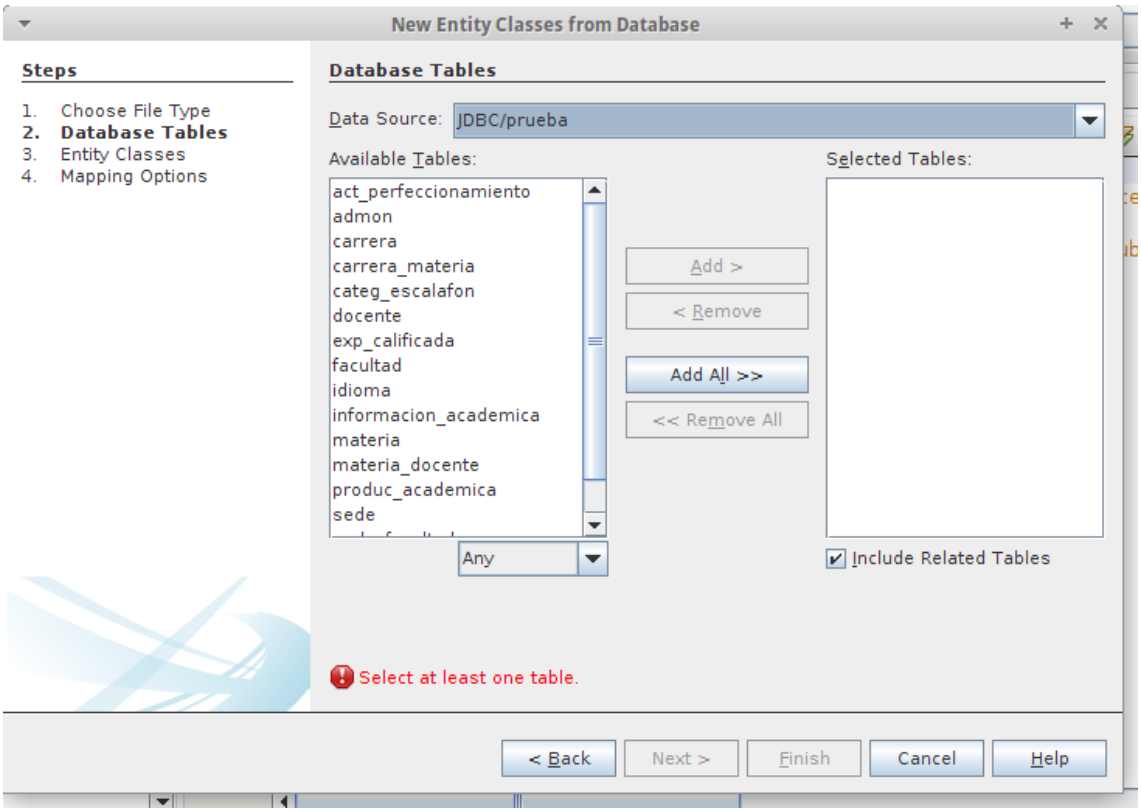
21. El siguiente paso es crear la persistencia pero con JPA.
    - En categorías seleccionamos persistence y en tipo de archivos entity Classes from Database



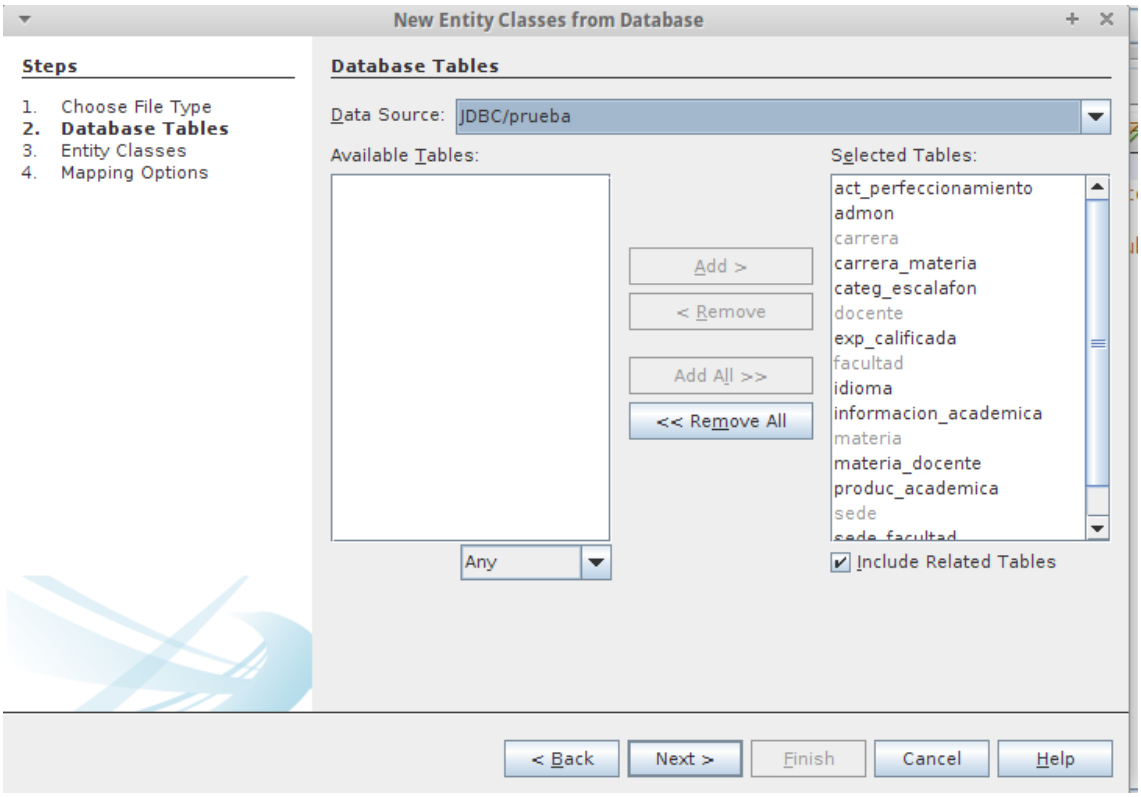
22. Creamos una conexión con la base de datos.



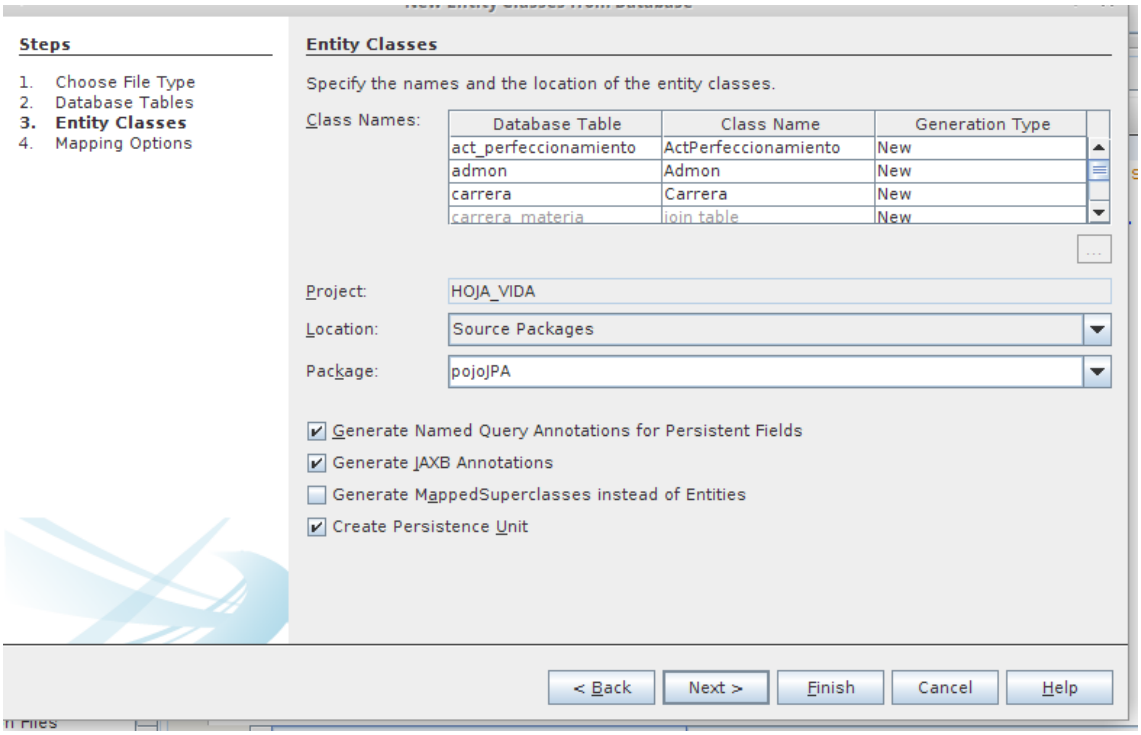
23. Nos aparecen las tablas de la base de datos



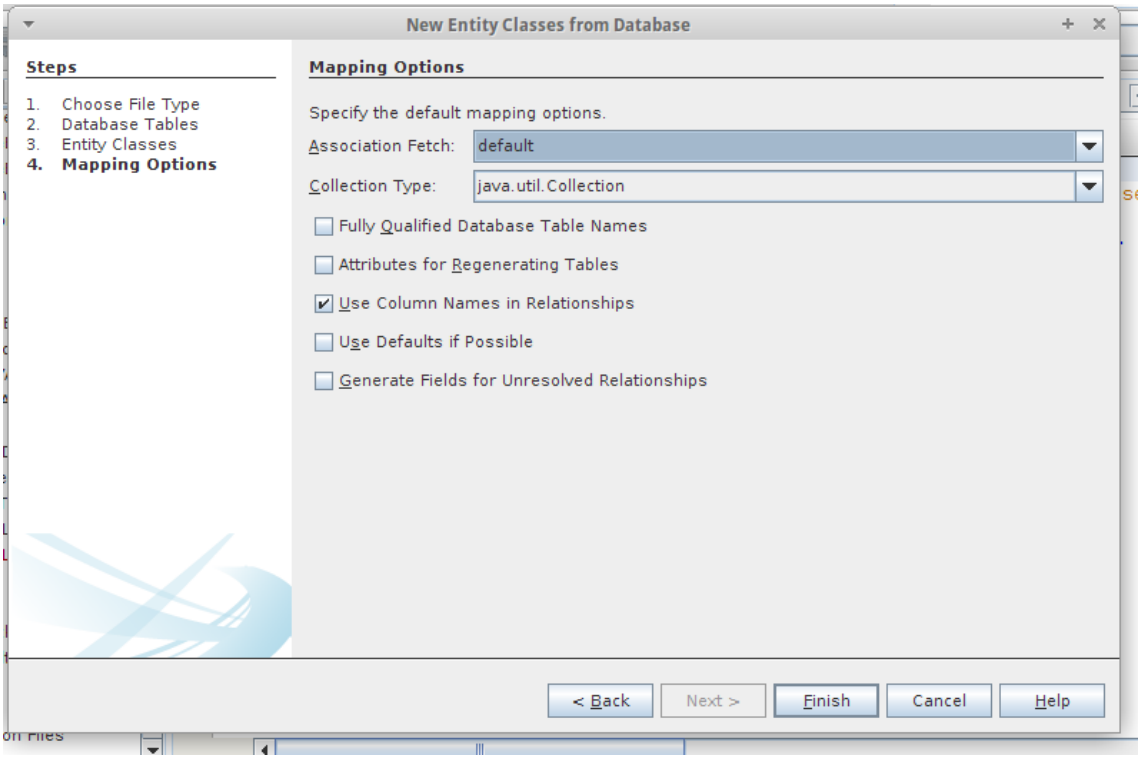
24. Seleccionamos todas las tablas



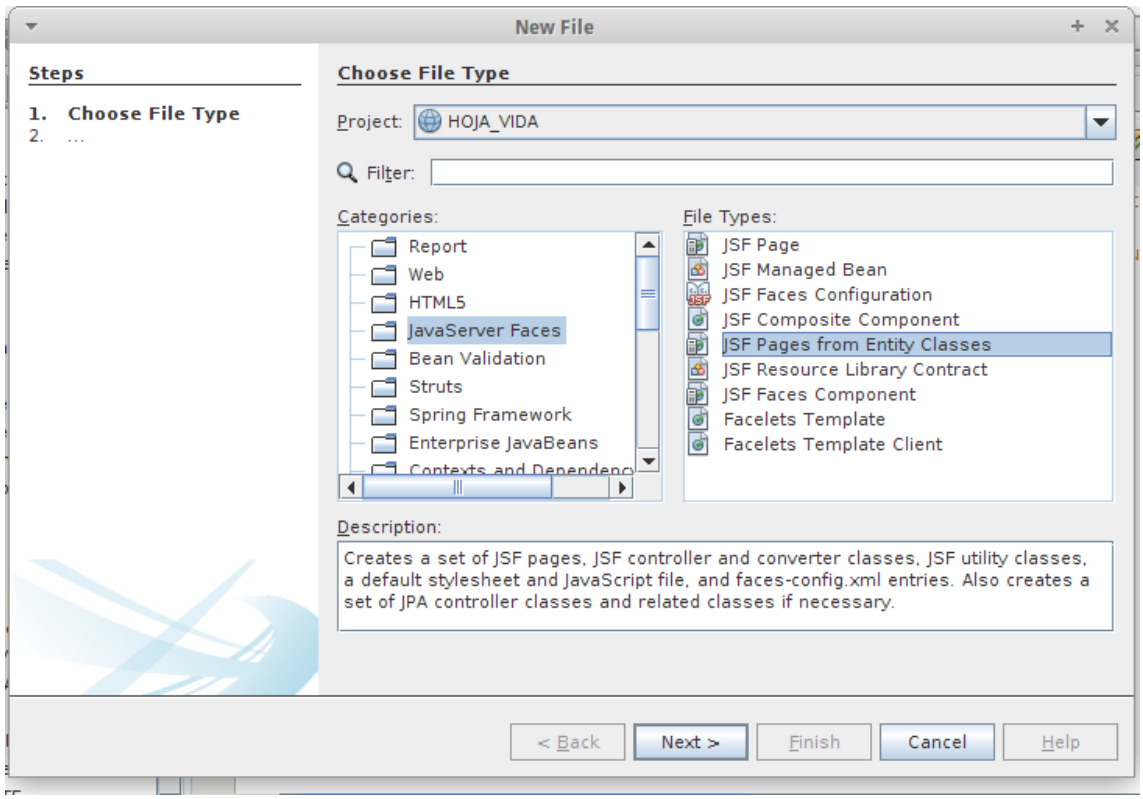
25. Creamos el paquete para los pojos de JPA



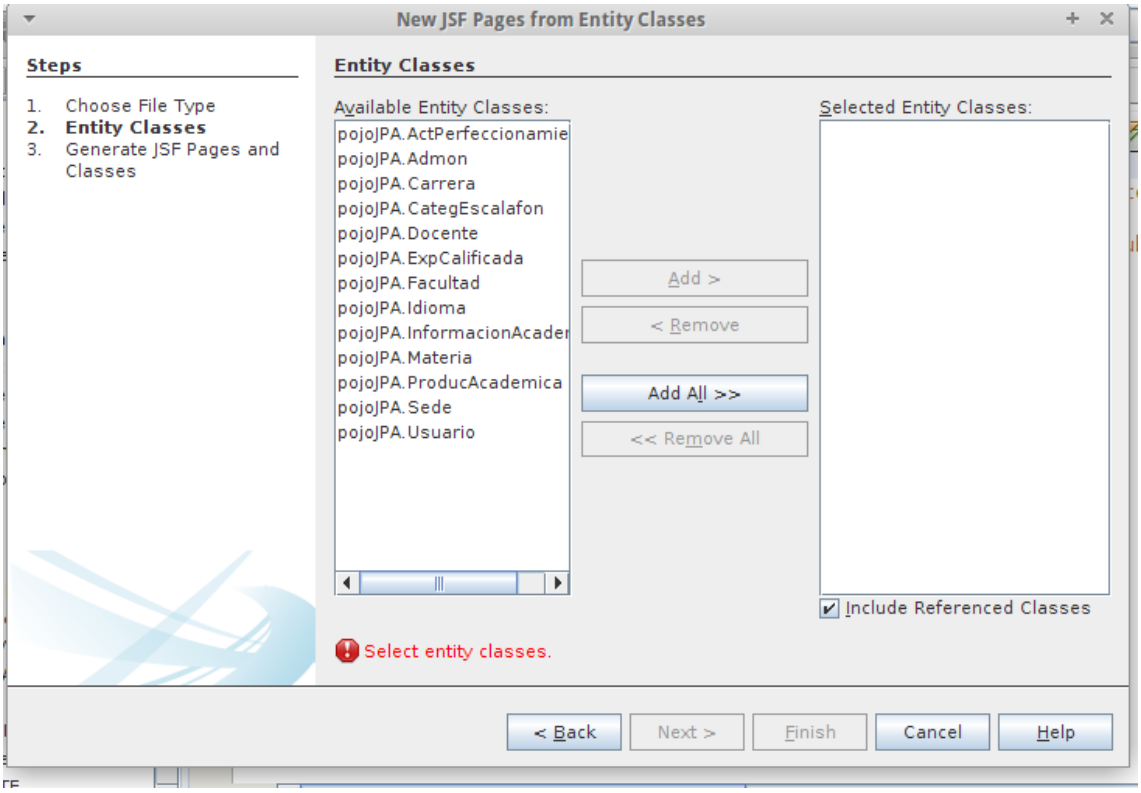
26. Para finalizar



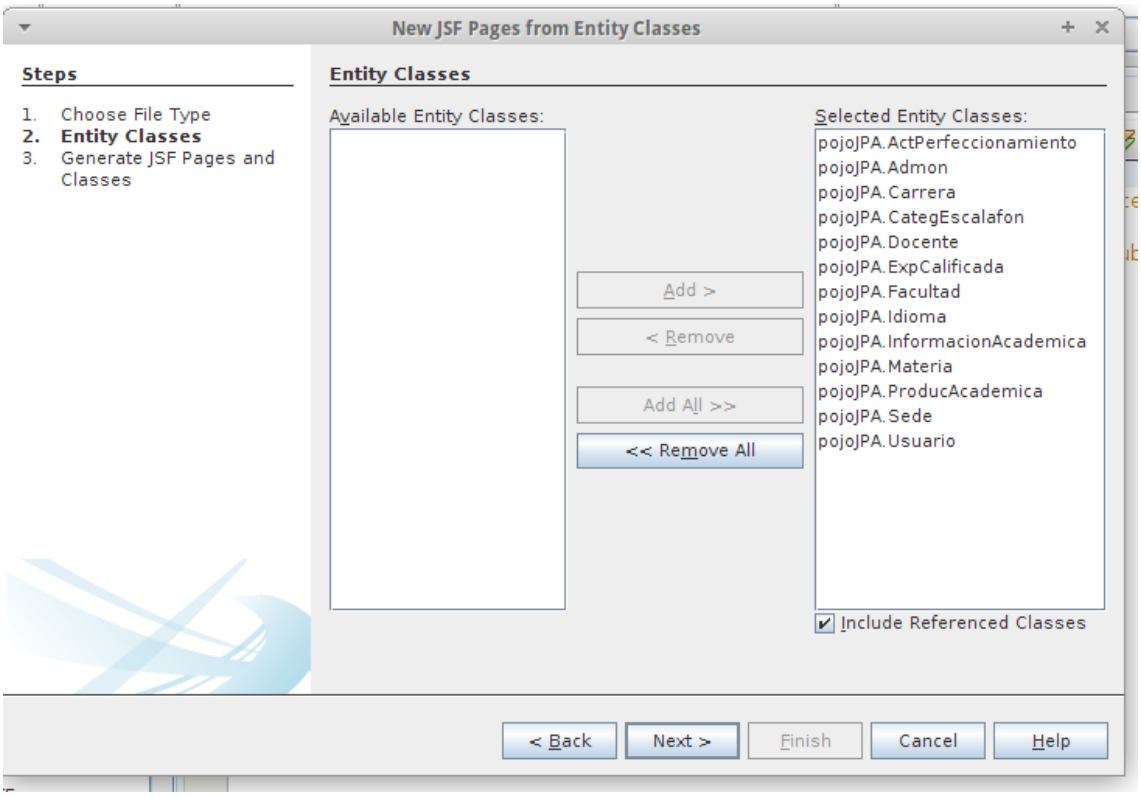
27. El siguiente paso es crear un archivo donde vamos a utilizar javaServer Faces y en tipo de archivos seleccionaremos JSF Pages from Entity Clases.



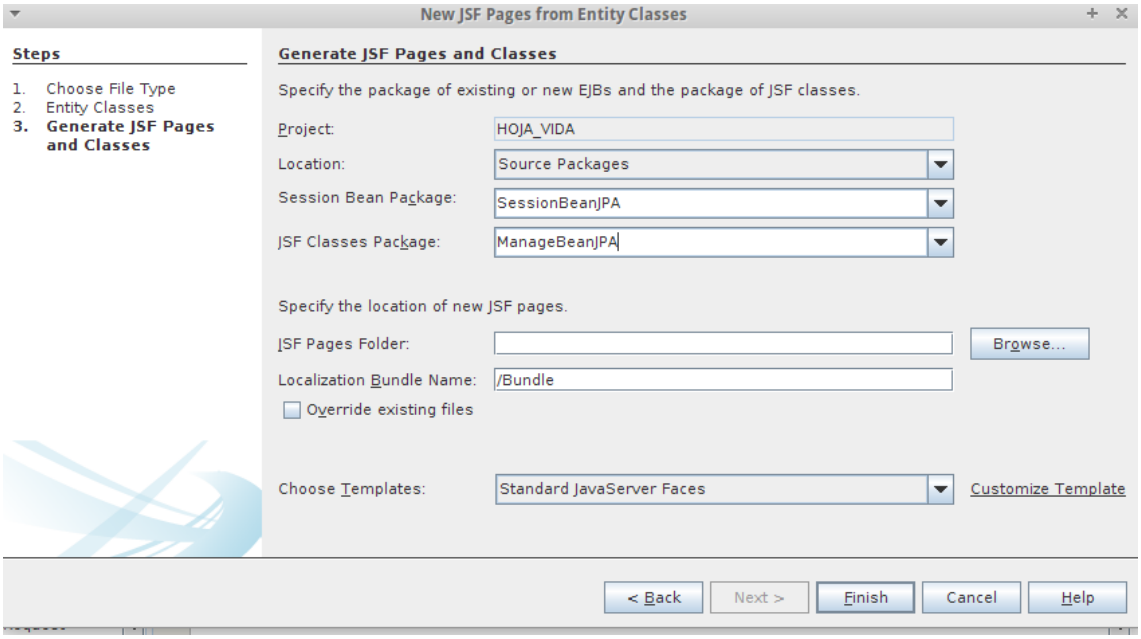
28. Nos aparece los pojos del JPA que creamos en los paso anteriores.



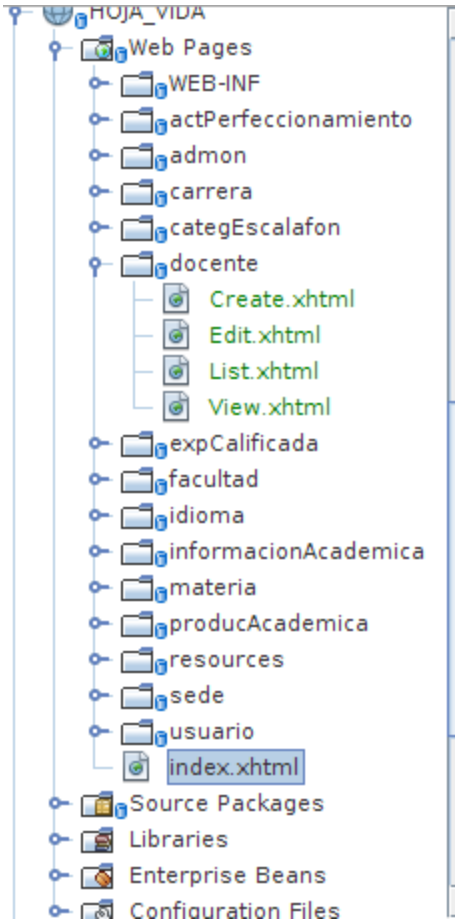
29. Seleccionamos todos los pojos JPA



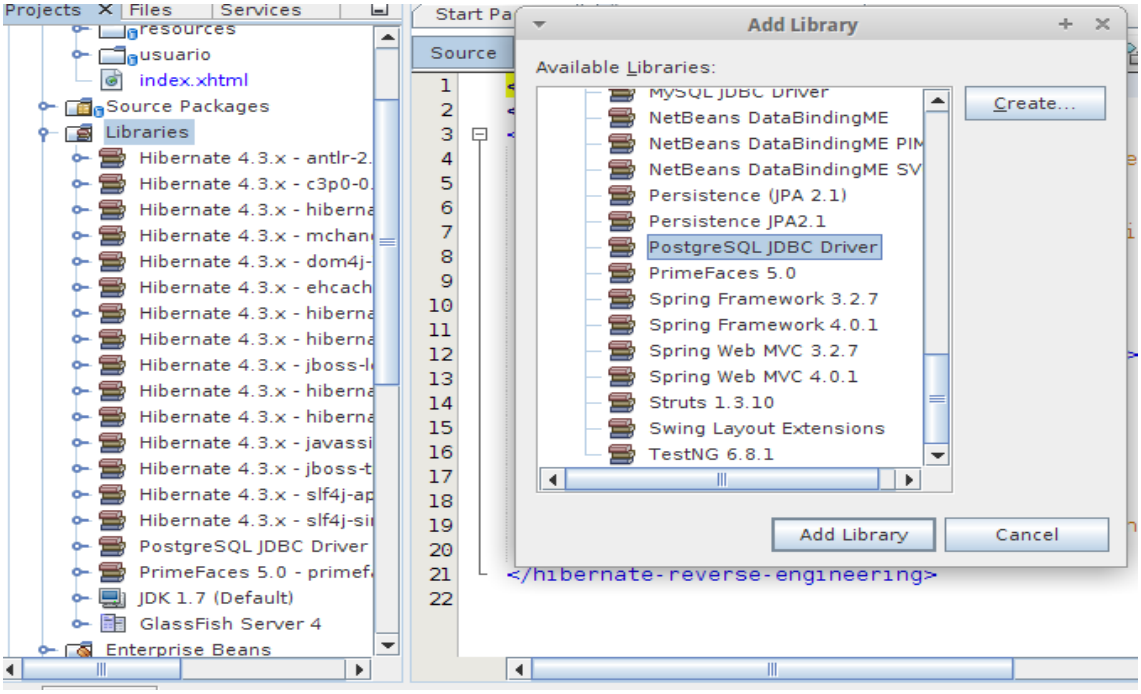
30. Creamos 2 carpetas . una para los sesión bean. Y la otra para la las clases jsf.



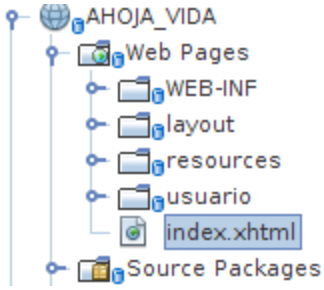
31. Se nos crearan en la parte de las vistas. Todos los formularios para crear, editar, listar, y ver. Para cada tabla.



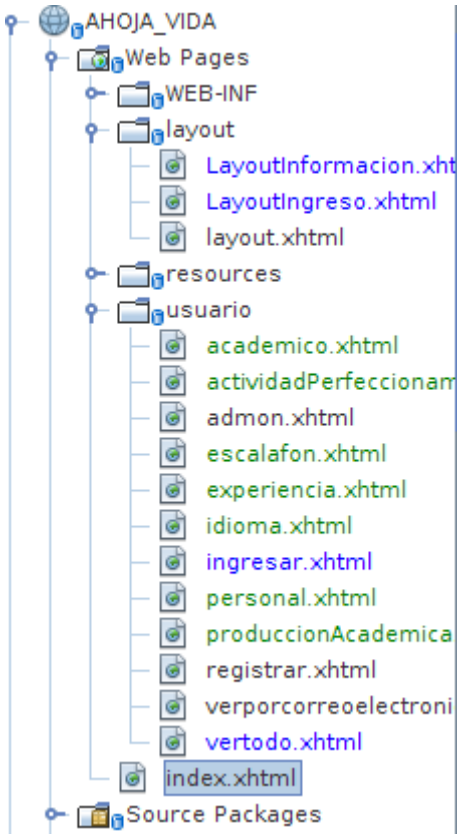
32. El siguiente paso y muy importante es colocar la librería de de postgres en el proyecto



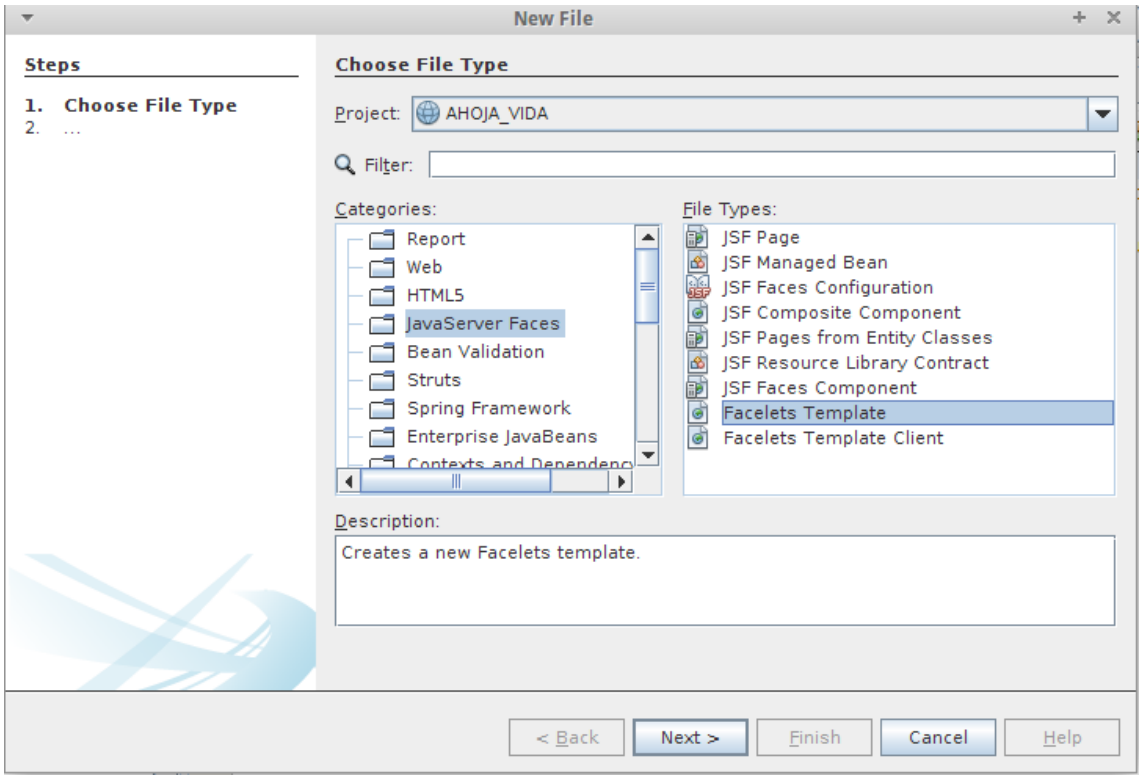
33. para la parte de las vistas .
    la estructura del proyecto queda de la siguiente manera.
    - Creamos una carpeta llamada layout donde se almacenaran los layout principales.
    - Creamos otra carpeta llamada usuario donde se almacenara los layout clientes, que dependen de los layout principales.

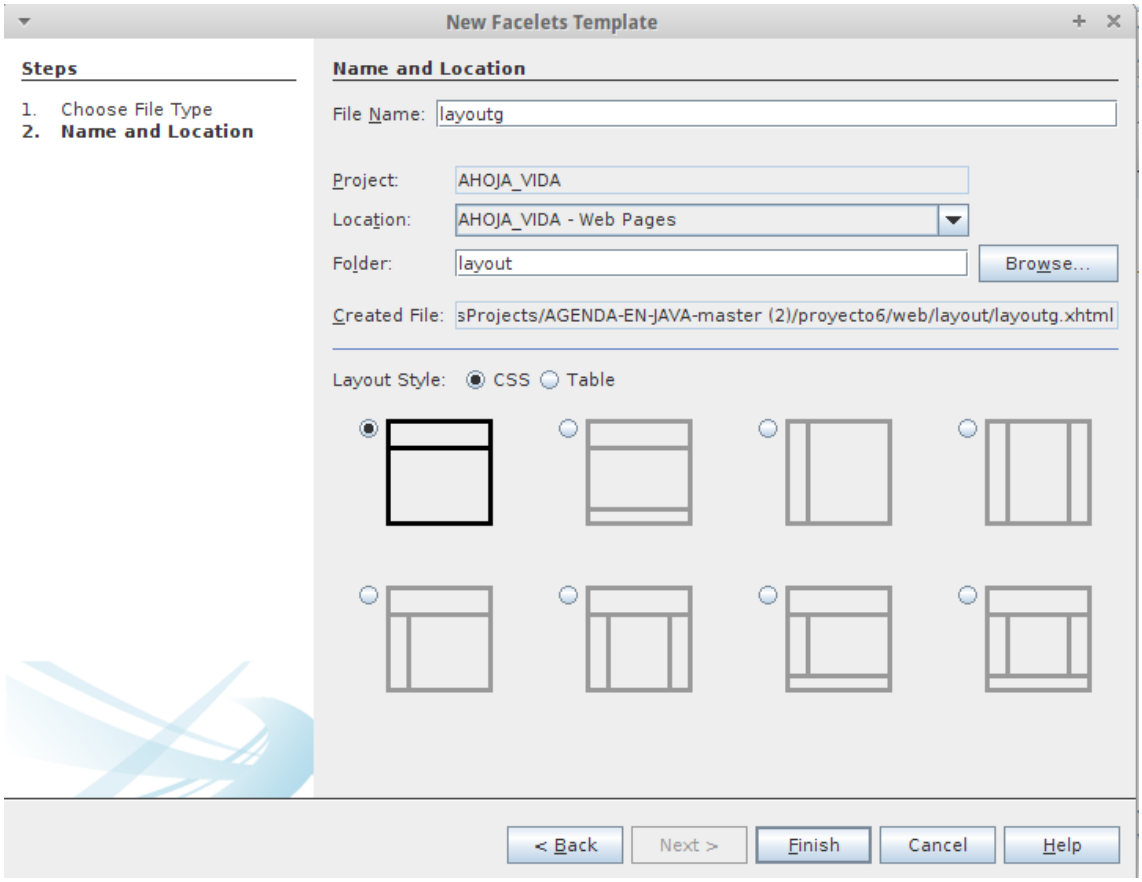


34. Las carpetas layout y usuario nos quedan de la siguiente manera:

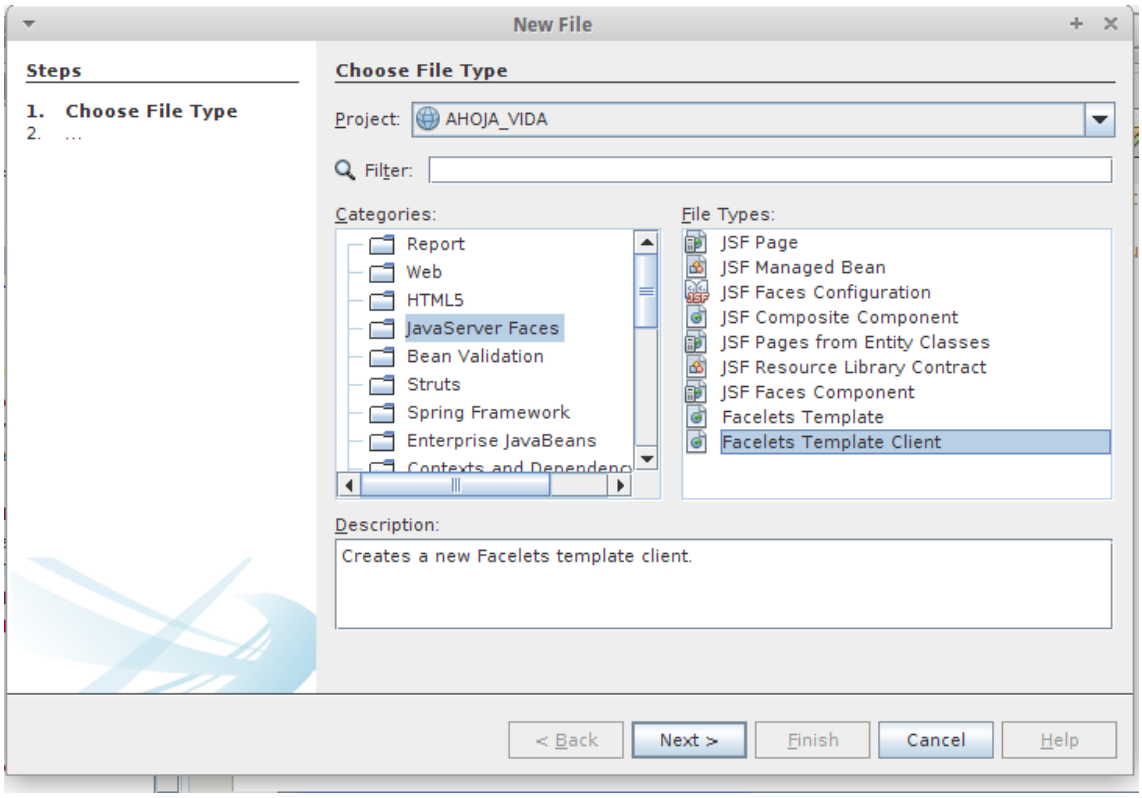


35. Para crear un layout principal lo que debemos hacer es

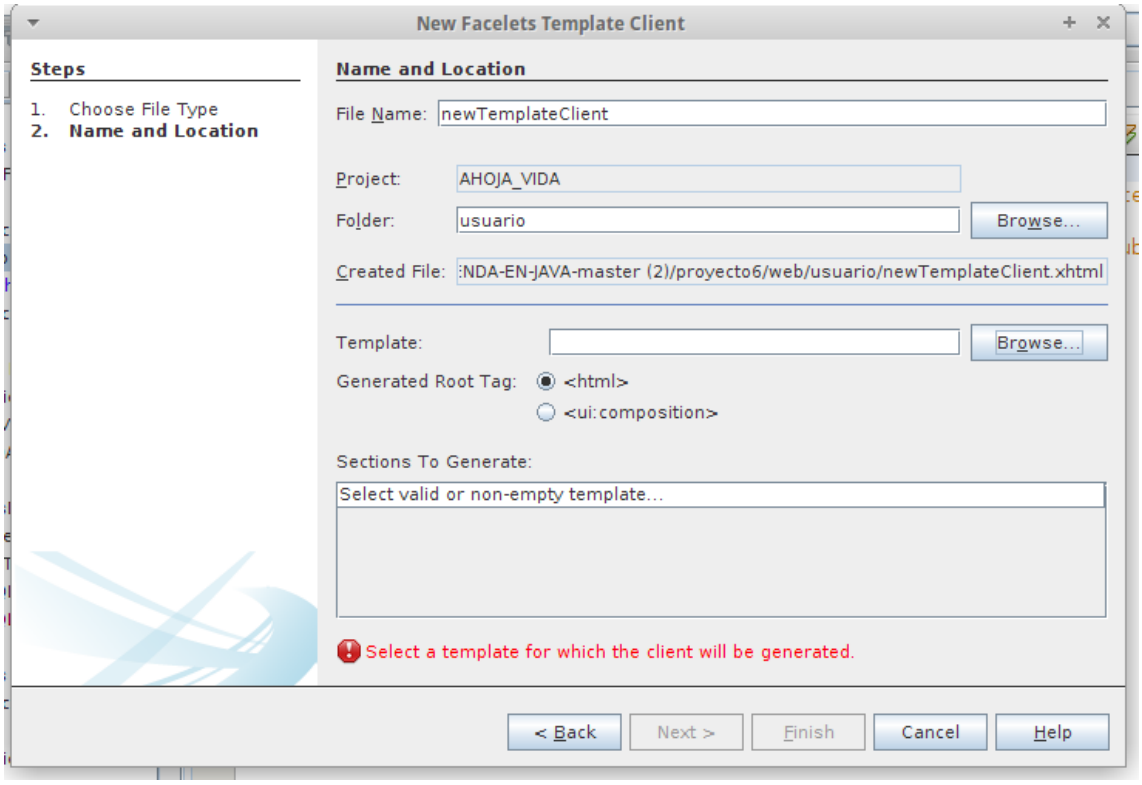




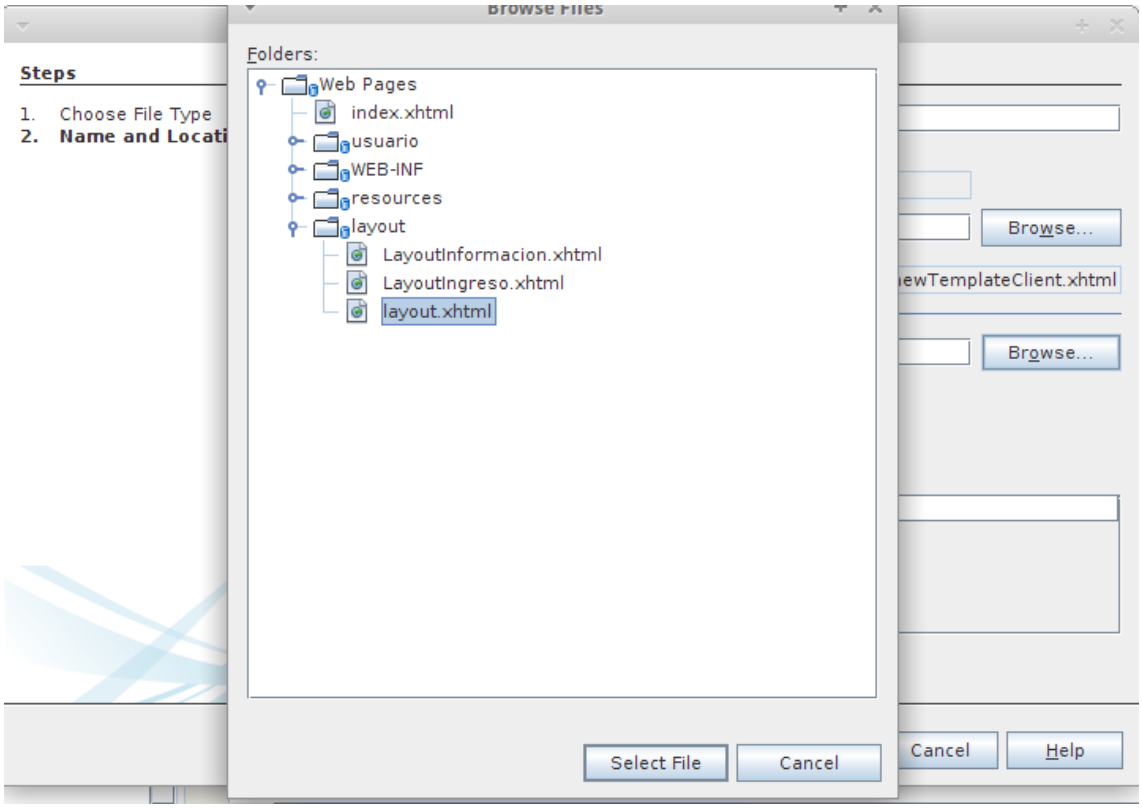
36. Y para crear los layout clientes debemos realizar lo siguiente



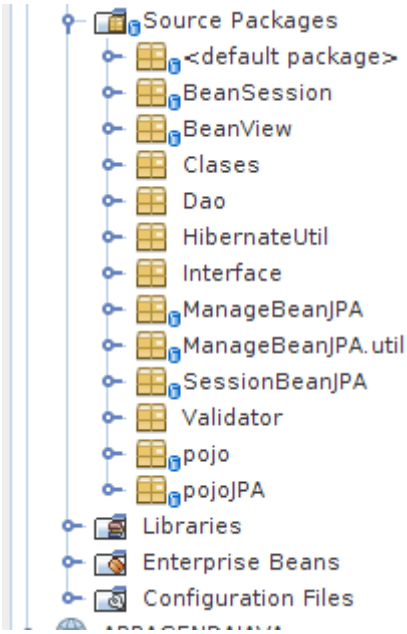
Oprimimos en la opción Browse para escoger el template general



Por ultimo seleccionamos nuestro template principal y seleccionamos finish.



37. La estructura de las carpetas del proyecto quedan de la siguiente forma.



6.2 Resultado final de la Aplicación

Resolución inicial, la cual se debe aceptar para seguir en el proceso.

UDEC UNIVERSIDAD DE CUNDINAMARCA

BANCO DE TALENTO HUMANO

Inicio

EL RECTOR DE LA UNIVERSIDAD DE CUNDINAMARCA, en uso de sus facultades legales y reglamentarias en especial las conferidas por el artículo 22 del Acuerdo 010 de 2.010, "Estatuto General" y

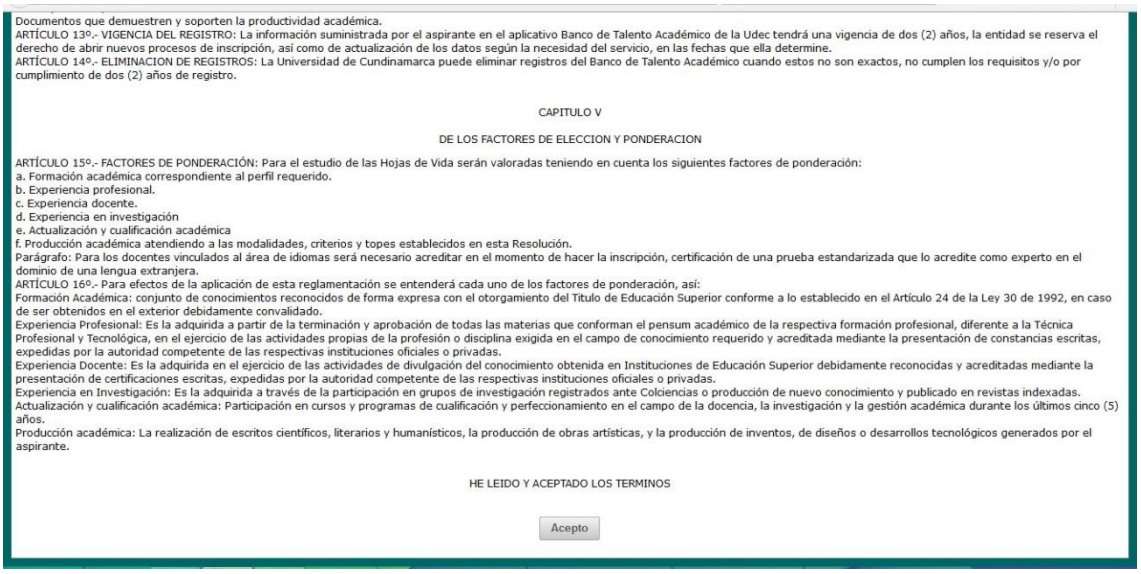
CONSIDERANDO

Que el artículo 69 de la Constitución Política garantiza la Autonomía Universitaria.
Que el artículo 3º de la Ley 30 de 1992, de conformidad con la Constitución Política de Colombia garantiza la Autonomía Universitaria.
Que las condiciones de vinculación de Docentes Ocasionales y Catedráticos, está regulada en los Artículos 73º y 74º de la Ley 30 de 1992; el Artículo 3º del Decreto 1279 de 2002; el Artículo 20º del Acuerdo 021 de 1993; los Acuerdos 060, 064 y 066 de 2002, y la Sentencia C-006 de 1996
La vinculación de los profesores ocasionales y catedráticos debe hacerse conforme a las reglas que defina cada universidad, con sujeción a lo dispuesto por la Ley 30 de 1992 y demás disposiciones constitucionales y legales vigentes.
Que el artículo 19, del Acuerdo 010 de 2002 "Estatuto General", establece que el Rector es el representante legal de la Universidad y como tal responsable de su dirección académica, administrativa, financiera, de bienestar Universitario, ordenador, nominador y ejecutor de las políticas y decisiones del Consejo Superior y del Consejo Académico.
Que es necesario reglamentar el Banco de Talento Académico del cual se seleccionan los docentes catedráticos y ocasionales que requieran los diferentes programas académicos para la ejecución de las actividades académicas.
Que por lo expuesto,

RESUELVE

ARTÍCULO 1º.- Reglamentar el procedimiento y funcionamiento del Banco de Talento Académico para la elección del personal académico requerido por la Universidad de Cundinamarca.

CAPITULO I



Documentos que demuestren y soporten la productividad académica.
ARTÍCULO 13º.- VIGENCIA DEL REGISTRO: La información suministrada por el aspirante en el aplicativo Banco de Talento Académico de la Udec tendrá una vigencia de dos (2) años, la entidad se reserva el derecho de abrir nuevos procesos de inscripción, así como de actualización de los datos según la necesidad del servicio, en las fechas que ella determine.
ARTÍCULO 14º.- ELIMINACION DE REGISTROS: La Universidad de Cundinamarca puede eliminar registros del Banco de Talento Académico cuando estos no son exactos, no cumplen los requisitos y/o por cumplimiento de dos (2) años de registro.

CAPITULO V

DE LOS FACTORES DE ELECCION Y PONDERACION

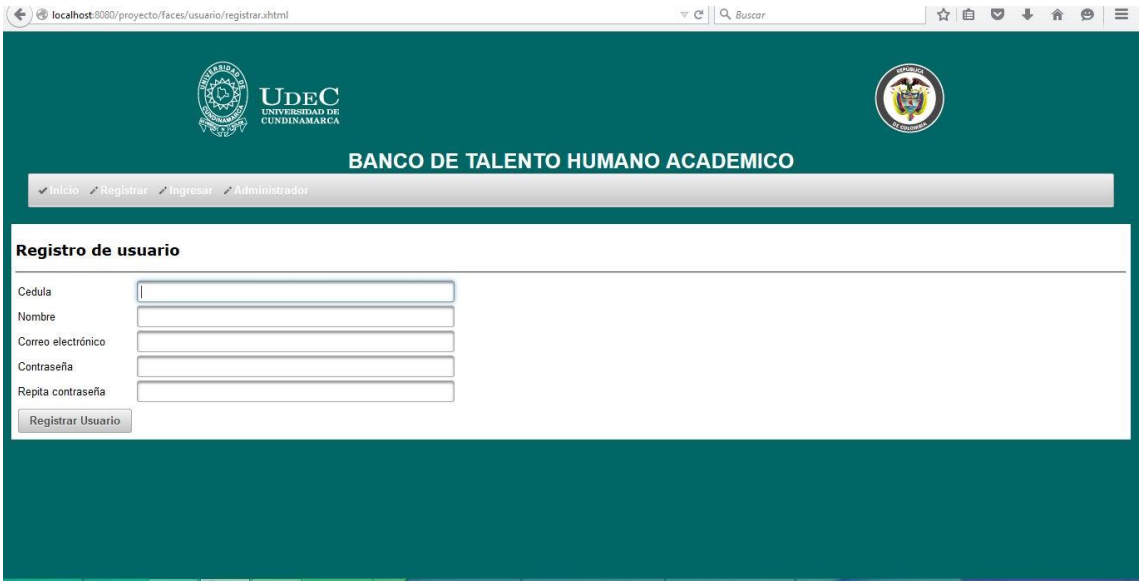
ARTÍCULO 15º.- FACTORES DE PONDERACIÓN: Para el estudio de las Hojas de Vida serán valoradas teniendo en cuenta los siguientes factores de ponderación:
a. Formación académica correspondiente al perfil requerido.
b. Experiencia profesional.
c. Experiencia docente.
d. Experiencia en investigación
e. Actualización y cualificación académica
f. Producción académica atendiendo a las modalidades, criterios y topes establecidos en esta Resolución.
Parágrafo: Para los docentes vinculados al área de idiomas será necesario acreditar en el momento de hacer la inscripción, certificación de una prueba estandarizada que lo acredite como experto en el dominio de una lengua extranjera.
ARTÍCULO 16º.- Para efectos de la aplicación de esta reglamentación se entenderá cada uno de los factores de ponderación, así:
Formación Académica: conjunto de conocimientos reconocidos de forma expresa con el otorgamiento del Título de Educación Superior conforme a lo establecido en el Artículo 24 de la Ley 30 de 1992, en caso de ser obtenidos en el exterior debidamente convalidado.
Experiencia Profesional: Es la adquirida a partir de la terminación y aprobación de todas las materias que conforman el pensum académico de la respectiva formación profesional, diferente a la Técnica Profesional y Tecnológica, en el ejercicio de las actividades propias de la profesión o disciplina exigida en el campo de conocimiento requerido y acreditada mediante la presentación de constancias escritas, expedidas por la autoridad competente de las respectivas instituciones oficiales o privadas.
Experiencia Docente: Es la adquirida en el ejercicio de las actividades de divulgación del conocimiento obtenida en Instituciones de Educación Superior debidamente reconocidas y acreditadas mediante la presentación de certificaciones escritas, expedidas por la autoridad competente de las respectivas instituciones oficiales o privadas.
Experiencia en Investigación: Es la adquirida a través de la participación en grupos de investigación registrados ante Colciencias o producción de nuevo conocimiento y publicado en revistas indexadas.
Actualización y cualificación académica: Participación en cursos y programas de cualificación y perfeccionamiento en el campo de la docencia, la investigación y la gestión académica durante los últimos cinco (5) años.
Producción académica: La realización de escritos científicos, literarios y humanísticos, la producción de obras artísticas, y la producción de inventos, de diseños o desarrollos tecnológicos generados por el aspirante.

HE LEIDO Y ACEPTADO LOS TERMINOS

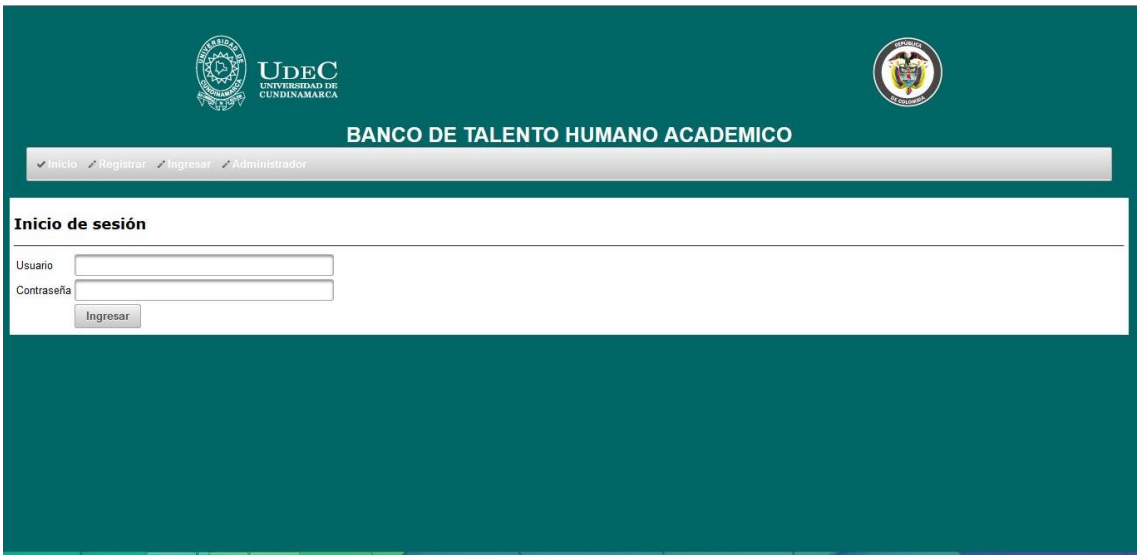
Acepto

Menú principal, donde se puede escoger cada uno de los módulos de la aplicación

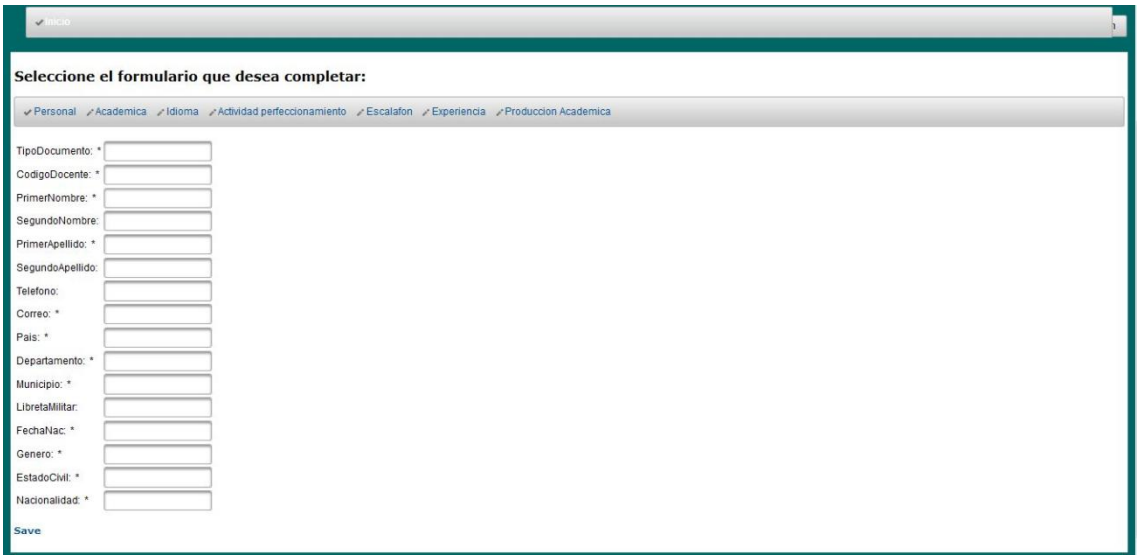
Interfaz inicial de registro de usuario



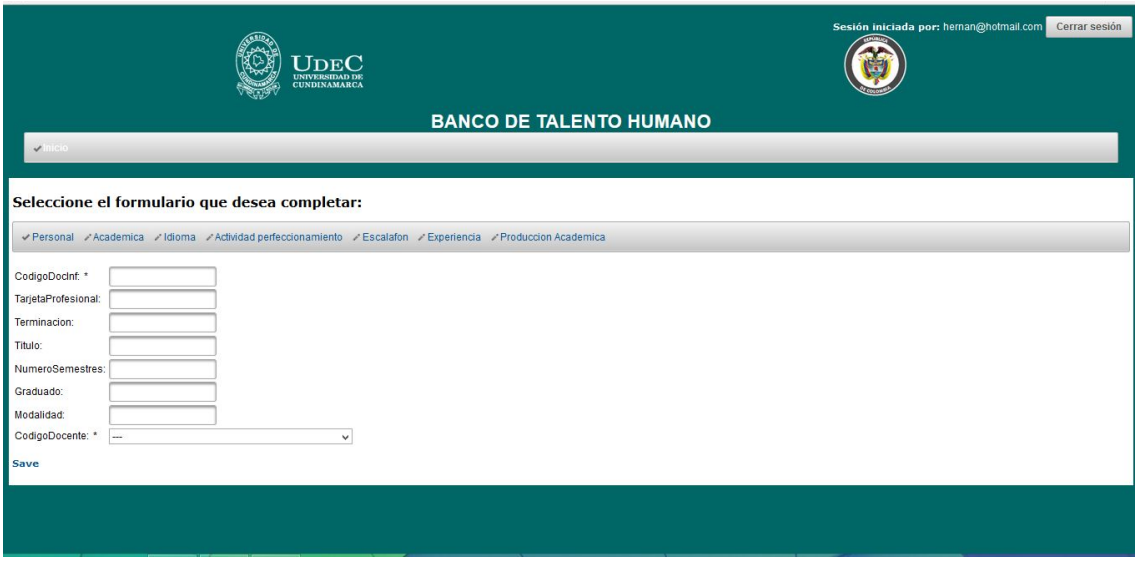
Interfaz de inicio de sesión según el rol del usuario (Administrador, docente)



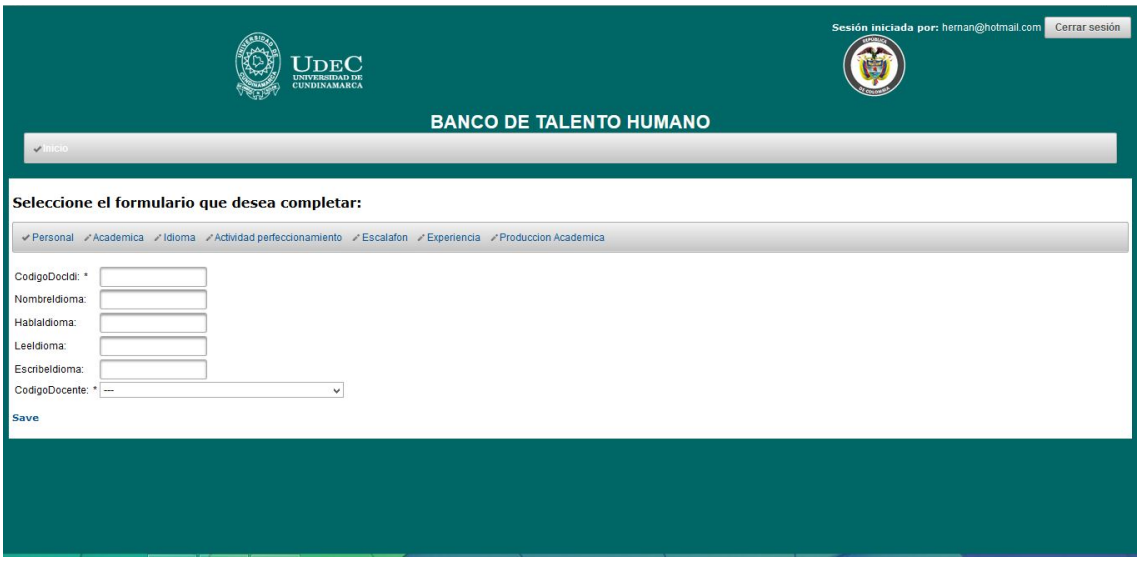
Módulo de ingreso de Información Básica



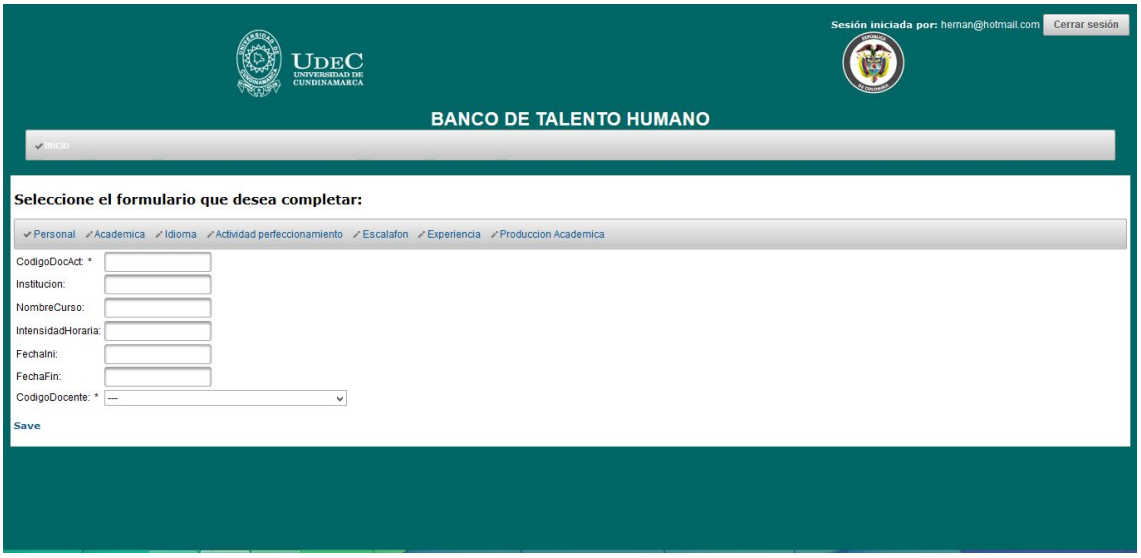
Módulo de Ingreso de Información académica



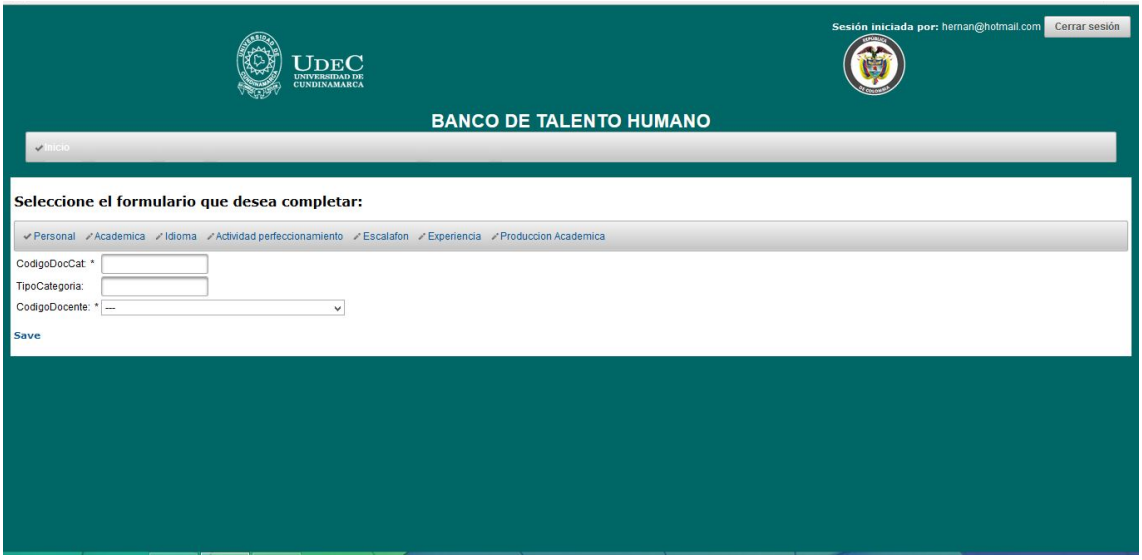
Módulo de ingreso de Idiomas



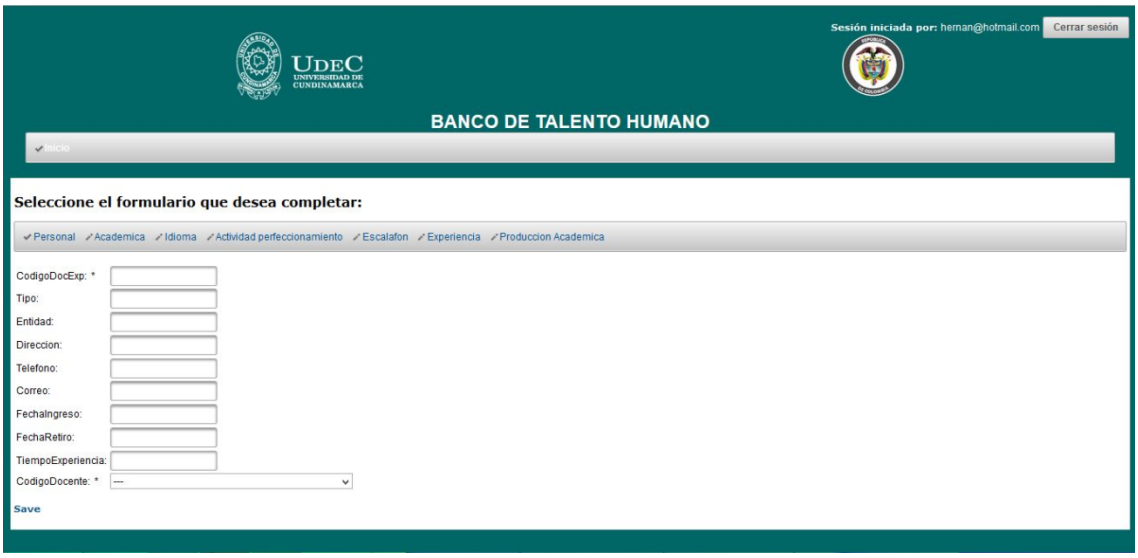
Modulo ingreso actividades de perfeccionamiento



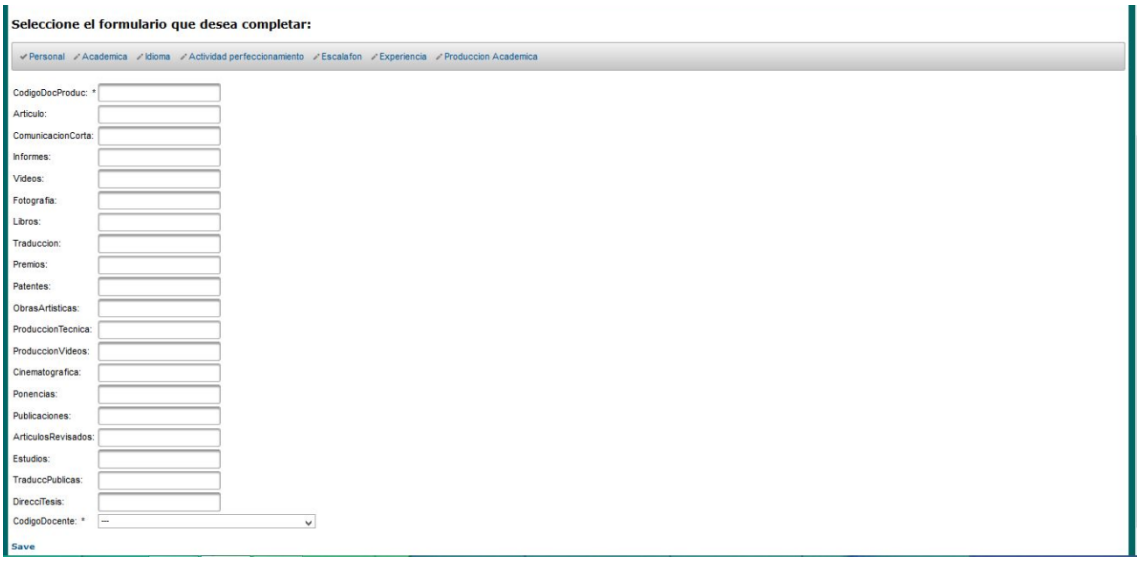
Modulo Ingreso categoría docente



Modulo ingreso Experiencia

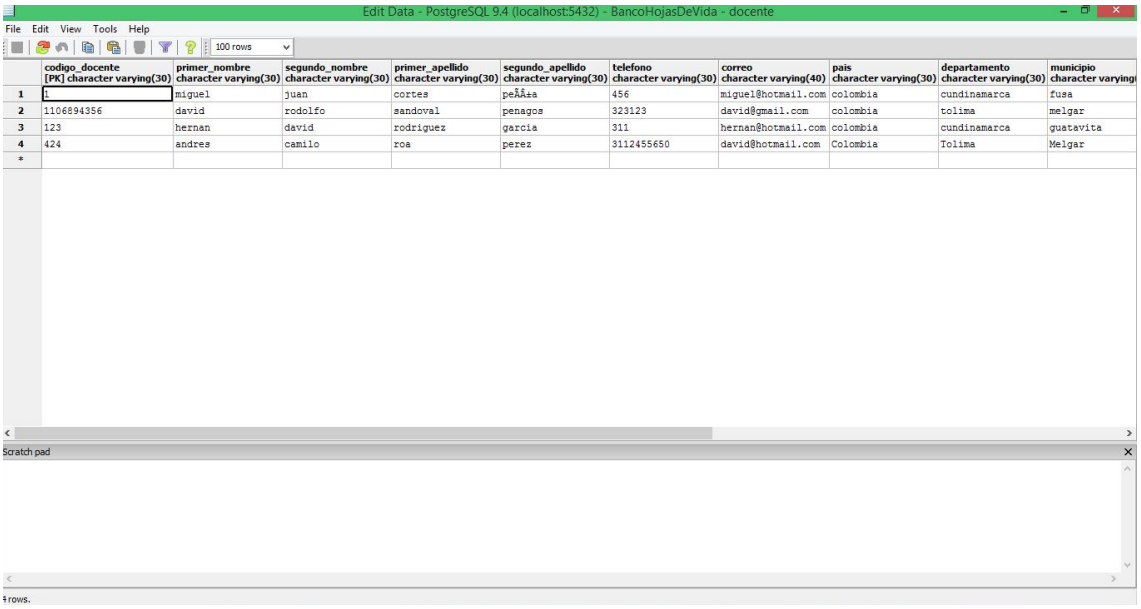


Modulo ingreso productividad académica



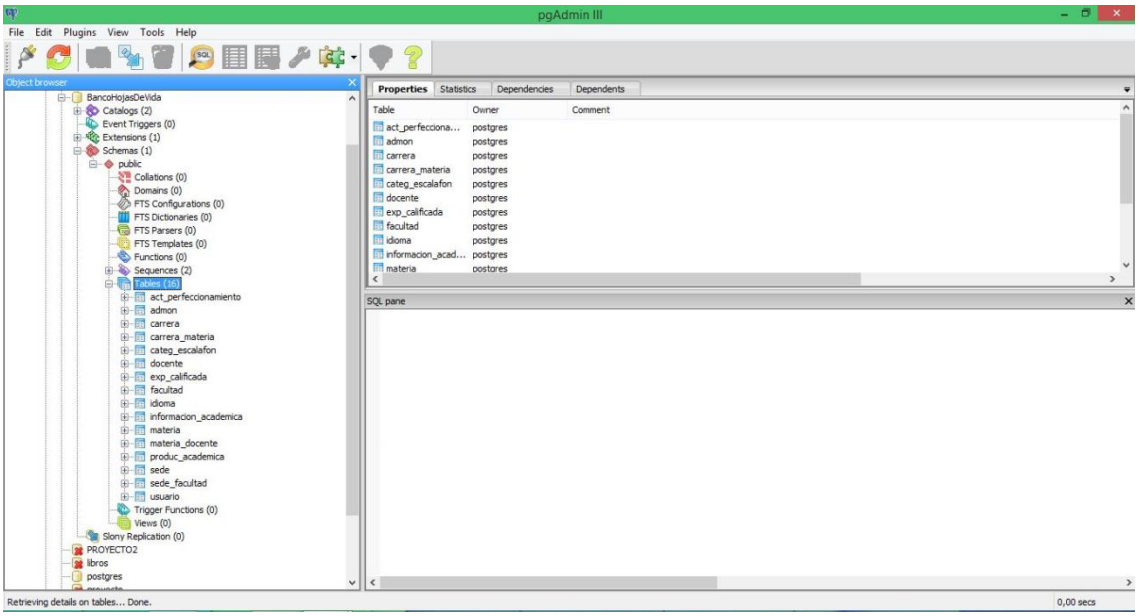
Modulo Información administrador

Pantalla de Postgresql con una consulta de Docentes registrados



	codigo_docente [PK] character varying(30)	primer_nombre character varying(30)	segundo_nombre character varying(30)	primer_apellido character varying(30)	segundo_apellido character varying(30)	telefono character varying(30)	correo character varying(40)	pais character varying(30)	departamento character varying(30)	municipio character varying
1	1	miguel	juan	cortes	peÃÃsa	456	miguel@hotmail.com	colombia	cundinamarca	fusa
2	1106894356	david	rodolfo	sandoval	penagos	323123	david@gmail.com	colombia	tolima	melgar
3	123	hernan	david	rodriguez	garcia	311	hernan@hotmail.com	colombia	cundinamarca	guatavita
4	424	andres	camilo	roa	perez	3112455650	david@hotmail.com	Colombia	Tolima	Melgar
*										

Pantalla de Postgresql con la base de datos BancoHojasDeVida



7 CONCLUSIONES

El sistema desarrollado permite el manejo de la información de los docentes de la Universidad de Cundinamarca, teniendo en cuenta tanto su información básica como de estudios realizados, idiomas, experiencia y publicaciones.

Para la realización del proyecto han sido muy útiles los conocimientos adquiridos en la maestría, ya que se han aplicado temas vistos como lo es la ingeniería de software en cuanto a las metodologías de desarrollo en especial XP, las bases de datos en cuanto al diseño del modelo entidad relación y el modelo relacional, este último creado sobre Postgresql, desarrollo de aplicaciones web en cuanto al manejo de servidores, xhtml, jsp, servlets.

La realización del proyecto fin de maestría nos da una visión más amplia de la aplicación del software libre en los diferentes ámbitos profesionales

El proyecto desarrollado puede ser mejorado y aplicado al manejo de hojas de vida no solo de docentes, sino también al personal que labora dentro de la universidad como lo es la parte administrativa.

7. BIBLIOGRAFÍA

Burns, E. (2010). *JavaServer Faces 2.0: The Complete Reference.* New York, Chicago, San Francisco, Lisboa, Londres, Madrid, Ciudad de México, Milán, Nueva Delhi, San Juan, Seúl, Singapur, Sydney, Toronto.

García, M. A. (s.f.). *JavaServer Pages.* Obtenido de http://crr.cacsae.com/java/servlets/Tutorial%20de%20JSP.pdf

ISOFT2010-2011. (2010). Obtenido de http://www.vc.ehu.es/jiwotvim/ISOFT2010-2011/Teoria/BloqueIV/Servlets.pdf

Piñol, C. M. (s.f.). *cibernetia*. Obtenido de http://www.cibernetia.com/manuales/introduccion_aplicaciones_web/2_0_www.php

Mateu, Carles**.** *Desarrollo de aplicaciones web*

http://materials.cv.uoc.edu/cdocent/NCF24UBLAGGISIWI3UC9.pdf

Aycard, Pérez David. Ingeniería de Software en entornos de software libre

http://*materials.cv.uoc.edu/cdocent/APK5H8_0MRUMW3K7YJKJ.pdf*