omissis

A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

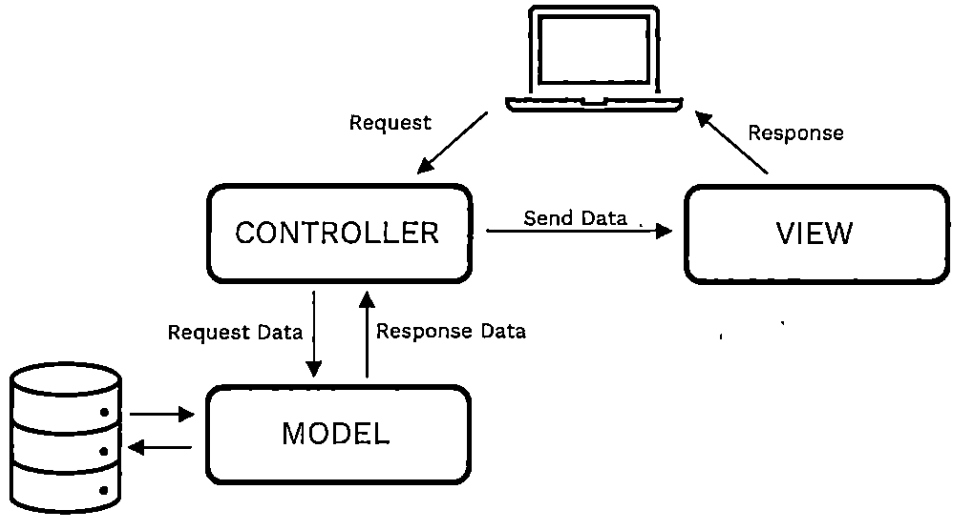
# QUESITO 1

JAVA: Descrivere la porzione di codice sorgente riportata di seguito

```
@Entity
@Audited
@Table(name = "richieste_acquisto", catalog = "ar_acquisti")
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(discriminatorType = DiscriminatorType.STRING, name = "acquisto_type_id", length = 15)
public abstract class RichiestaAcquisto implements Serializable {
	private static final long serialVersionUID = -4957228965079426816L;

	@Id
	@GeneratedValue(strategy = IDENTITY)
	@Column(name = "id", nullable = false)
	private Integer id;

	@NotNull
	@ManyToOne (optional = false)
	@JoinColumn(name = "id_richiedente", nullable = false)
	private Utente richiedente;

	@NotNull
	@Column(name = "oggetto", nullable = false, length = 255)
	private String oggetto;

	@Lob
	@NotNull
	@Column(name = "ragioni_acquisto", nullable = false)
	private String ragioniAcquisto;

	@NotNull
	@Column(name = "spesa_prevista_budget", nullable = false)
	private boolean spesaPrevistaBudget = true;

	@ManyToOne (optional = true)
	@JoinColumn (name = "id_fornitore", nullable = true)
	private Fornitore fornitore;

	@OneToMany(mappedBy = "richiestaAcquisto", cascade = CascadeType.ALL)
	private Set<Prodotto> prodotti = new HashSet<Prodotto>();

	@Lob
	@Column(name = "note")
	private String note;

	@SortNatural
	@OneToMany(mappedBy = "richiestaAcquisto", cascade = CascadeType.ALL)
	private SortedSet<StatoRichiestaAcquisto> stati = new TreeSet<>();

	@OneToOne(optional = true, cascade = CascadeType.ALL)
	@JoinColumn(name = "id_stato_corrente")
	private StatoRichiestaAcquisto statoCorrente;

	@Transient
	public Set<Bene> getBeni() {
		return getProdotti().stream().filter(p -> p instanceof Bene)
			.map(p -> (Bene) p).collect(Collectors.toSet());
	}
	@Transient
	public Set<Servizio> getServizi() {
		return getProdotti().stream().filter(p -> p instanceof Servizio)
			.map(p -> (Servizio) p).collect(Collectors.toSet());
	}
}
```

A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

DESIGN PATTERN: Nel design pattern MVC di cosa si occupa la componente *Model*?




LINUX: Che cosa fanno i comandi `ls`, `cd` e `pwd` nella shell?

VCS: Che cos'è un *branch* e perché è utile nei sistemi di controllo versione?

GDPR: Liceità del trattamento (art. 6)

STATUTO: Il Rettore. Chi è e quali sono le principali funzioni. (art. 10)

INGLESE:
Spring Data's mission is to provide a familiar and consistent, Spring-based programming model for data access while still retaining the special traits of the underlying data store.
It makes it easy to use data access technologies, relational and non-relational databases, map-reduce frameworks, and cloud-based data services. This is an umbrella project which contains many subprojects that are specific to a given database. The projects are developed by working together with many of the companies and developers that are behind these exciting technologies.

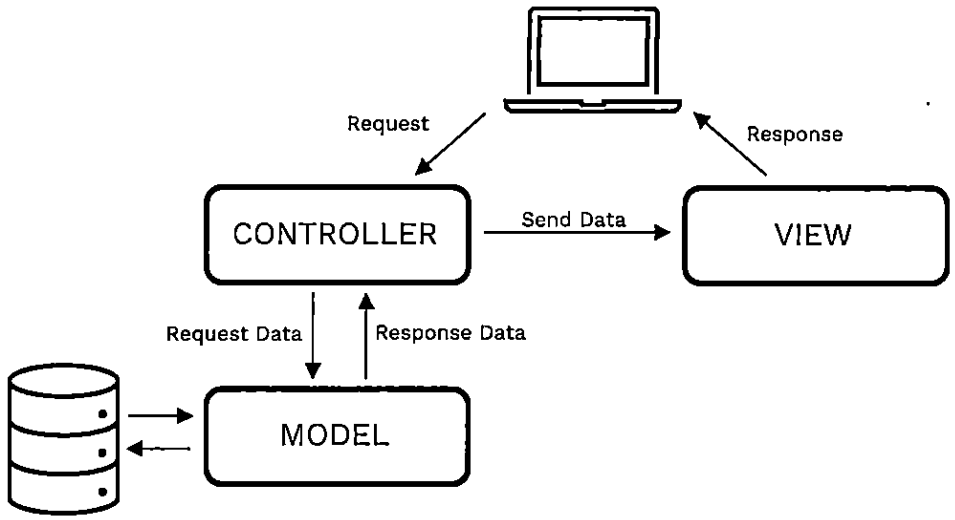
# QUESITO 2

JAVA: Descrivere la porzione di codice sorgente riportata di seguito

```
@Entity
@Audited
@Table(name = "prodotti", catalog = "ar_acquisti")
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(discriminatorType = DiscriminatorType.STRING, name = "prodotto_type_id", length = 15)
public abstract class Prodotto implements Serializable {
	private static final long serialVersionUID = 8044706214954496539L;

	@Id
	@GeneratedValue(strategy = IDENTITY)
	@Column(name = "id", nullable = false)
	private Integer id;

	@NotNull
	@ManyToOne (optional = false)
	@JoinColumn (name = "id_richiesta_acquisto", nullable = false)
	protected RichiestaAcquisto richiestaAcquisto;

	@NotNull
	@Column(name = "nome", nullable = false, length = 255)
	private String nome;

	@NotNull
	@Column(name = "quantita", nullable = false)
	@Min(1)
	private int quantita = 0;

	@Column(name = "importo_unitario_previsto")
	private Double importoUnitarioPrevisto;

	@NotNull
	@Column(name = "liquidazione_in_valuta_estera", nullable = false)
	private boolean liquidazioneInValutaEstera = false;


	@Transient
	private String getLocalCurrencyFromDouble(double money) {
		NumberFormat formatter = NumberFormat.getCurrencyInstance(Locale.ITALY);
		return formatter.format(money);
	}

	@Transient
	public String getImportoUnitarioPrevistoCurrency() {
		return getLocalCurrencyFromDouble(getImportoUnitarioPrevisto());
	}

	@Transient
	public Double getImportoTotalePrevisto() {
		return getImportoUnitarioPrevisto()*getQuantita();
	}

	@Transient
	public String getImportoTotalePrevistoCurrency() {
		return getLocalCurrencyFromDouble(getImportoTotalePrevisto ());
	}


}
```

A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

DESIGN PATTERN: Nel design pattern MVC di cosa si occupa la componente *View*?




LINUX: Qual è la differenza tra un utente normale e l'utente root in un sistema operativo POSIX?

VCS: Qual è la differenza tra un sistema di controllo versione centralizzato e uno distribuito?

GDPR: Diritto di accesso dell'interessato (art. 15)

STATUTO: Il Direttore Generale. Chi è e quali sono le principali funzioni. (art. 24)

INGLESE:
Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications.
Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements.

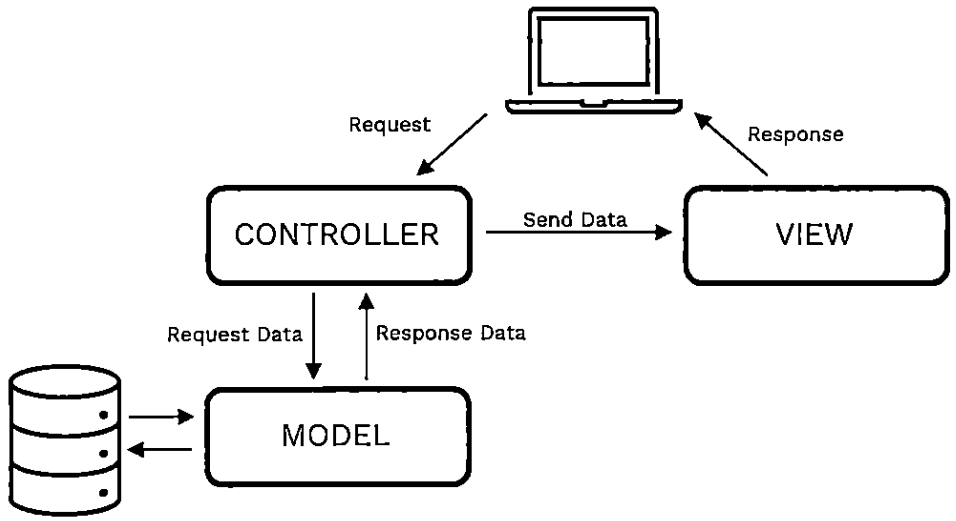
# QUESITO 3

JAVA: Descrivere la porzione di codice sorgente riportata di seguito

```
@Entity
@Audited
@Table(name = "richiesta_acquisto_rdo_mepa", catalog = "ar_acquisti")
@DiscriminatorValue("RDO_MEPA")
public class RichiestaAcquistoRdoMepa extends RichiestaAcquisto {
	private static final long serialVersionUID = 5690916037022169978L;

	@NotNull
	@Enumerated(EnumType.STRING)
	@Column(name = "tipo_rdo", length = 25)
	private TipoRDO tipoRDO = TipoRDO.CONFRONTO_PREVENTIVI;

	@NotNull
	@ManyToOne
	@JoinColumn(name = "id_persona_fisica_rup", nullable = false)
	private PersonaFisica responsabileUnicoProcedimento; // RUP

	@ManyToMany
	@JoinTable(
		name = "richieste_acquisto_rdo_mepa_fornitori_invitati",
		catalog = "ar_acquisti",
		joinColumns = @JoinColumn (name = "id_richiesta_acquisto"),
		inverseJoinColumns = @JoinColumn (name = "id_fornitore")
	)
	protected Set<Fornitore> fornitori = new HashSet<>();

	@NotNull
	@Column(name = "importo_totale_presunto", nullable = false)
	private Double importoTotalePresunto;

	@NotNull
	@Column(name = "liquidazione_in_valuta_estera", nullable = false)
	private boolean liquidazioneInValutaEstera = false;

	public RichiestaAcquistoRdoMepa() {
		super();
	}

	@Transient
	public String getImportoTotalePresuntoCurrency() {
		if (getImportoTotalePresunto() != null) {
			double money = getImportoTotalePresunto();
			NumberFormat formatter = NumberFormat.getCurrencyInstance(Locale.ITALY);
			return formatter.format(money);
		}
		return "0";
	}
}
public enum TipoRDO {

	TRATTATIVA_DIRETTA ("Trattativa diretta", true),
	CONFRONTO_PREVENTIVI "Confronto di preventivi", false),
	RDO_SEMPLICE "Richiesta di Offerta semplice", true);

	private final String descrizione;
	private final boolean unicoFornitoreConvolto;

	private TipoRDO(String descrizione, boolean unicoFornitoreCoinvolto) {
		this.descrizione = descrizione;
		this. unicoFornitoreConvolto = unicoFornitoreCoinvolto;
	}
}
```

A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

DESIGN PATTERN: Nel design pattern MVC di cosa si occupa la componente *Controller*?




LINUX: Che cosa sono i permessi dei file in GNU/Linux e come si leggono dalla stringa visualizzata dal comando `ls -l`?

VCS: Che cosa rappresenta un *tag* nei sistemi di controllo versione e quando può essere utile crearlo?

GDPR: Protezione dei dati fin dalla progettazione e protezione dei dati per impostazione predefinita (*data protection by default and by design*) (art. 25)

STATUTO: Il Senato Accademico. Composizione e principali funzioni. (artt. 16 e 17)

INGLELSE:
Spring Data JPA, part of the larger Spring Data family, makes it easy to easily implement JPA-based (Java Persistence API) repositories. It makes it easier to build Spring-powered applications that use data access technologies.
Spring Data JPA aims to significantly improve the implementation of data access layers by reducing the effort to the amount that's actually needed. As a developer you write your repository interfaces using any number of techniques, and Spring will wire it up for you automatically. You can even use custom finders or query by example and Spring will write the query for you!

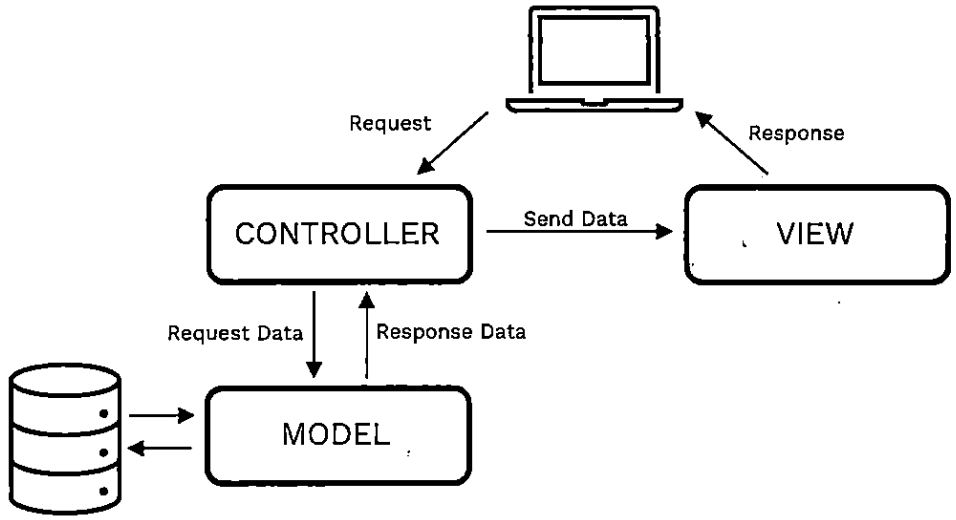
# QUESITO 4

JAVA: Descrivere la porzione di codice sorgente riportata di seguito

```
@Entity
@Audited
@Table(name = "fornitori", catalog = "ar_acquisti")
public class Fornitore implements Serializable {
	private static final long serialVersionUID = 6996386577912102800L;

	@Id
	@GeneratedValue(strategy = IDENTITY)
	@Column(name = "id", nullable = false)
	private Integer id;

	@NotEmpty
	@Column(name = "ragione_sociale", nullable = false, length = 255)
	private String ragioneSociale;

	@Pattern (regexp = "([0-9]{11})?", message = "Partita IVA non valida")
	@Column(name = "partita_iva", length = 11)
	private String partitaIVA;

	@Column(name = "vat_number", length = 11)
	private String vatNumber;

	@Email (message = "Formato non corretto per l'indirizzo e-mail")
	@Column(name = "email", length = 255)
	private String email;

	@Email (message = "Formato non corretto per la posta elettronica certificata")
	@Column(name = "pec", length = 255)
	private String pec;

	@Lob
	@Column(name = "note", nullable = true)
	private String note;

	@NotNull
	@Column(name = "attivo", nullable = false)
	private boolean attivo = true;

	@OneToMany(mappedBy = "fornitore")
	private Set<RichiestaAcquisto> richiesteAcquisto = new HashSet<RichiestaAcquisto>();

	@ManyToMany (mappedBy = "fornitori")
	private Set<RichiestaAcquisto> richiesteAcquistoRdoMepa = new HashSet<RichiestaAcquisto>();

	@Transient
	public String getIdentifications () {
		String cod = "";
		cod += (StringUtils.isNotBlank(getPartitaIVA()) ? ("PIVA: " + getPartitaIVA() + "; ") : "";
		cod += (StringUtils.isNotBlank(getVatNumber()) ? ("VAT N: " + getVatNumber () + "; ") : "";
		return code;
	}

}
```

A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

DESIGN PATTERN: A cosa serve l'adozione del design pattern MVC?




LINUX: Che cosa contiene generalmente la directory `/etc` in un sistema GNU/Linux?

VCS: Cosa rappresenta l'operazione di *commit* nei sistemi di controllo versione?

GDPR: Responsabile della protezione dei dati (RDO o DPO). Designazione, posizione e compiti (artt. 37, 38 e 39)

STATUTO: Il Consiglio di Amministrazione. Composizione e principali funzioni. (artt. 20 e 21)

INGLESE:
Envers is an extension to Hibernate ORM that provides an easy way to add auditing / versioning for entities.
Envers provides:

- auditing for all O/R mappings defined by the JPA specification, along with some mappings specific to Hibernate ORM,
- logged change data for each revision using a revision entity, and
- querying for historical snapshots of an entity and its associations.

Much like source code version control, Envers uses the concept of revisions. A revision identifies the full set of changes to audited entity fields which occurred within a given transaction.