```
@startuml

class User {
  userId : int [PK]
  profilePhoto : img
  name : string
  surname : string
  phoneNumber : string
  linkedIn : string
  email : string
  password : string
  securityQuestion : string
  securityAnswer : string
  biography : text
  role : enum { student, academicTutor, companyTutor }
  isActive : boolean
  isVerified : boolean
  preferredLanguage : string
  institutionId : int
  department : string
  internalRole : string
  certifications : text
  spokenLanguages : text
  messageNotificationsEnabled : boolean
  internshipNotificationsEnabled : boolean
  calendarNotificationsEnabled : boolean
  theme : enum { LIGHT, DARK }
  --
  login()
  register()
  recoverPassword()
  changeLanguage()
  interactWithAssistant()
  viewHomePage()
  updateProfile()
  deleteAccount()
}

class Institution {
  institutionId : int [PK]
  logo : string
  name : string
  address : string
  contactEmail : string
```

```
  contactNumber : string
  sector : string
  size : string
  description : text
  isVerified : boolean
  studentDomains : text
  tutorDomains : text
  isDeletable : boolean
  --
  addDomain(domainName : string)
  verify()
  updateInstitutionProfile()
  deleteInstitution()
  transferInstitutionOwnership()
}

class Domain {
  domainId : int [PK]
  domainName : string
  institutionId : int (FK)
  isVerified : boolean
  --
  verify(domainId : int)
}

class CV {
  cvId : int [PK]
  userId : int (FK)
  fileName : string
  fileType : string
  fileSize : int
  fileUrl : string
  --
  upload()
}

class PasswordRecoveryRequest {
  requestId : int [PK]
  userId : int (FK)
  requestDate : datetime
  status : enum { open, closed }
  --
  createRequest(email : string)
  closeRequest(requestId : int)
```

```
}

class AssistantChat {
  chatId : int [PK]
  userId : int (FK)
  startTime : datetime
  endTime : datetime
  chatLog : text (JSON)
  --
  startChat(userId : int)
  saveChat(chatLog : text)
}



class StageOffer {
  offerId : int [PK]
  title : string
  description : text
  category : enum { Marketing, Programming, HumanResources, Communication, Finance,
Design, Other }
  requirements : text
  duration : string
  compensation : string
  location : string
  deadline : date
  companyTutorId : int (FK on User, role=companyTutor)
  status : enum { DRAFT, PUBLISHED }
  lastModified : datetime
  requiredLanguages : text
  --
  createOffer()
  editOffer()
  publishOffer()
  saveDraft()
  deleteOffer()
  optimizeWithLLM()
}

class Match {
  matchId : int [PK]
  studentUserId : int (FK on User, role=student)
  stageOfferId : int (FK on StageOffer)
  matchPercentage : float
```

```
  feedback : enum { UP, DOWN, NONE }
  --
  sendOfferToStudent()
  setFeedback(feedback : enum)
}

class MeetingQuestionnaire {
  meetingQuestionnaireId : int [PK]
  selectionProcessId : int (FK on SelectionProcess)
  filledByUserId : int (FK on User, role=companyTutor)
  creationDate : datetime
  submissionDate : datetime (nullable if draft)
  answers : text (JSON)
  --
  fillQuestionnaire(data : struct)
  submit()
  viewQuestionnaire()
  editQuestionnaire()
  deleteQuestionnaire()
}

class FinalEvaluationQuestionnaire {
  finalEvalQuestionnaireId : int [PK]
  activeStageId : int (FK on ActiveStage)
  filledByUserId : int (FK on User)
  roleFiller : enum { STUDENT, COMPANY_TUTOR, ACADEMIC_TUTOR }
  creationDate : datetime
  submissionDate : datetime (nullable if draft)
  answers : text (JSON)
  --
  fillQuestionnaire(data : struct)
  submit()
  viewQuestionnaire()
  editQuestionnaire()
  deleteQuestionnaire()
}

class StageStateHistory {
  historyId : int [PK]
  activeStageId : int (FK on ActiveStage)
  state : enum { START_STAGE, SCHEDULED_EVENT, PAUSE, LAST_EVENT,
FINAL_EVALUATION, END_STAGE }
  startDate : datetime
  endDate : datetime (nullable if ongoing)
```

```
  description : text
  --
  changeState(newState : enum)
  setStartDate(...)
  setEndDate(...)
}

class Review {
  reviewId : int [PK]
  activeStageId : int (FK on ActiveStage)
  reviewerId : int (FK on User)
  reviewedUserId : int (FK on User)
  rating : int
  text : text
  creationDate : datetime
  --
  createReview(...)
  editReview(...)
  deleteReview()
  getAverageRating(userId : int)
}

class Notification {
  notificationId : int [PK]
  userId : int (FK)
  type : enum { CALENDAR_EVENT, INTERNSHIP_UPDATE, ISSUE }
  message : text
  date : datetime
  priority : enum { LOW, MEDIUM, HIGH }
  isRead : boolean
  --
  markAsRead()
  getShortcutLink() : string
}

class CalendarEvent {
  eventId : int [PK]
  creatorUserId : int (FK to User)
  title : string
  description : text
  eventDateStart : datetime
  eventDateEnd : datetime
  allDay : boolean
  location : string
```

```
  videoCallLink : string
  category : enum { MEETING, FEEDBACK, DELIVERY, OTHER }
  notifyBefore : int
  isFinalEvent : boolean
  isDeleted : boolean
  --
  createEvent()
  editEvent()
  deleteEvent()
  addParticipant(userId : int)
  removeParticipant(userId : int)
}

class CalendarEventParticipant {
  eventParticipantId : int [PK]
  eventId : int (FK on CalendarEvent)
  userId : int (FK on User)
  status : enum { PENDING, CONFIRMED, DECLINED }
  joinedDate : datetime
  --
  confirmPresence()
  declinePresence()
}

class Chat {
  chatId : int [PK]
  name : string
  description : string
  chatType : enum { REGULAR, PROBLEM, VIDEO }
  problemTitle : string
  problemCategory : enum { COMMUNICATION, TECHNICAL_SKILLS, TIME_MANAGEMENT,
INTERPERSONAL_PROBLEMS, OTHER }
  problemDescription : text
  problemStatus : enum { IN_PROGRESS, WAITING_FOR_RESPONSE, RESOLVED }
  videoCallDateTime : datetime
  creationDate : datetime
  createdByUserId : int (FK on User)
  activeStageId : int (FK on ActiveStage, optional)
  isDeleted : boolean
  --
  createChat(name : string, description : string, chatType : enum)
  addParticipant(userId : int)
  removeParticipant(userId : int)
  reportProblem(...)
```

```
  updateProblemStatus(newStatus : enum)
  scheduleVideoCall(dateTime : datetime)
  deleteChat()
}

class ChatParticipant {
  chatParticipantId : int [PK]
  chatId : int (FK on Chat)
  userId : int (FK on User)
  canAddOthers : boolean
  roleInChat : enum { CREATOR, MEMBER }
  joinedDate : datetime
  --
  assignRole(...)
  removeParticipant(...)
}

class Message {
  messageId : int [PK]
  chatId : int (FK on Chat)
  senderUserId : int (FK on User)
  content : text
  contentType : enum { TEXT, FILE, AUDIO, SYSTEM }
  timestamp : datetime
  isImportant : boolean
  isRead : boolean
  --
  sendMessage(senderId : int, content : text)
  markAsRead()
  attachFile(fileData : binary)
  highlightAsProblem()
}

class ActiveStage {
  activeStageId : int [PK]
  stageOfferId : int (FK on StageOffer)
  studentId : int (FK on User, role=student)
  academicTutorId : int (FK on User, role=academicTutor)
  companyTutorId : int (FK on User, role=companyTutor)
  startDate : datetime
  endDate : datetime
  status : enum { IN_PROGRESS, COMPLETED, SUSPENDED, TERMINATED }
  problematic : text
  --
```

```
  updateStageStatus(newStatus : enum)
  reportProblem(details : string)
  openChatForIssue()
  closeStage()
  suspendStage()
  resumeStage()
}

class SelectionProcess {
  selectionProcessId : int [PK]
  stageOfferId : int (FK on StageOffer)
  studentId : int (FK on User, role=student)
  academicTutorId : int (FK on User, role=academicTutor, optional)
  companyTutorId : int (FK on User, role=companyTutor)
  tag : enum { RECEIVED, SENT }
  status : enum { MATCHMAKING, NON_VISTO, VISTO, RIFIUTATO1, RIFIUTATO2,
ACCEPTED_STUDENT, ACCEPTED_COMPANY, MEETING_SET,
QUESTIONARIO_COMPANY, DECISION_STUDENT, STUDENT_RIFIUTA,
STUDENT_CALL_AGAIN, STUDENT_ACCETTA, TUTOR_ACADEMICO_SCELTO,
TUTOR_ACADEMICO_RIFIUTATO, DECISIONE_FINALE_AZIENDA,
PROCESS_CONCLUDED }
  deadline : date
  creationDate : datetime
  --
  acceptOffer(by : string)
  rejectOffer(by : string)
  scheduleMeeting(dateTime : datetime)
  fillQuestionnaire()
  chooseAcademicTutor(tutorEmail : string)
  finalDecisionCompany()
  removeProcess()
}

User "0..*" --> "1" Institution : "belongs to"
User "1" --> "0..1" CV : "has"
User "1" --> "0..*" PasswordRecoveryRequest : "requests"
User "1" --> "0..*" AssistantChat : "initiates"
Institution "1" *-- "0..*" Domain : "owns"

User "1" --> "0..*" StageOffer : "creates"
User "1" --> "0..*" Match : "matches"
User "1" --> "0..*" SelectionProcess : "participates in"
User "1*" --> "0..*" MeetingQuestionnaire : "fills"
User "1*" --> "0..*" FinalEvaluationQuestionnaire : "evaluates"
```

```
User "1" --> "0..*" ActiveStage : "participates in"
User "1" --> "0..*" Review : "writes"

StageOffer "1" --> "0..*" Match : "matches with"
StageOffer "1*" --> "0..*" SelectionProcess : "involves"
StageOffer "1*" --> "1" ActiveStage : "leads to"

SelectionProcess "1" --> "1" MeetingQuestionnaire : "produces"

FinalEvaluationQuestionnaire "0..*" --> "1" ActiveStage : "evaluates"

ActiveStage "1" --> "0..*" StageStateHistory : "has state"
ActiveStage "1" --> "0..*" Review : "evaluates"

User "1" *-- "0..*" Notification : "receives"
User "1" --> "0..*" CalendarEvent : "creates"
User "1" --> "0..*" CalendarEventParticipant : "belongs"
User "1" --> "0..*" ChatParticipant : "belongs"
User "1" --> "0..*" Message : "sends"
User "1" --> "0..*" Chat: "creates"

CalendarEvent "1" --> "0..*" CalendarEventParticipant : "has participants"

Chat "1" --> "0..*" ChatParticipant : "has participants"
Chat "0..*" --> "1" Message : "includes messages"
Chat "0..*" --> "0..1" ActiveStage : "linked to stage"
Chat "0..*" --> "0..1" SelectionProcess : "linked to stage"

@enduml
```