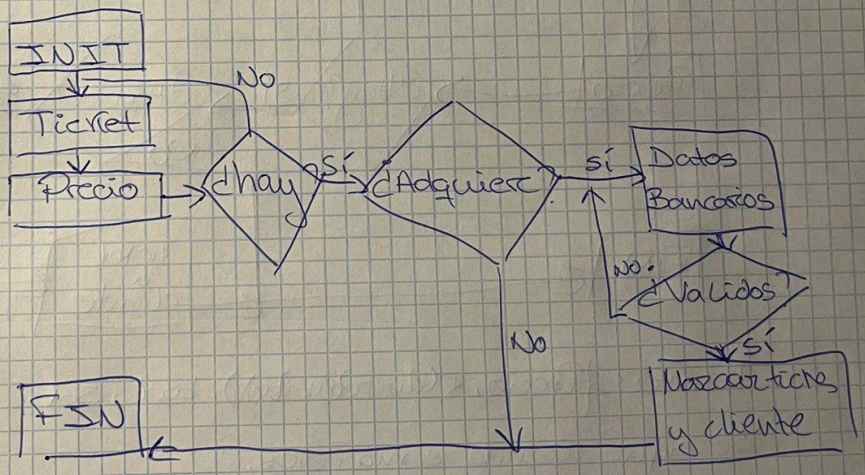
① a) El modelo se basa en cliente-servidor, en el cual el cliente (threads) que actuan de forma simultánea reservan los tickets en el servidor.

b) Código.

```
int sd;
int cd;
struct sockaddr serverSock clientSock;
bzero ((char *) &serverSock, sizeof(servSock));
serverSock->sin_family = AF_INET;
serverSock->sin_port = htons(??);
serverSock->sin_addr->s_addr = INADDR_ANY
bind(sd(struct sockaddr*)&serverSock, sizeof(serverSock));
listen(sd, tam);
cd = accept(sockAddr *) &clientSock, sizeof(clientsize));
```

② a)



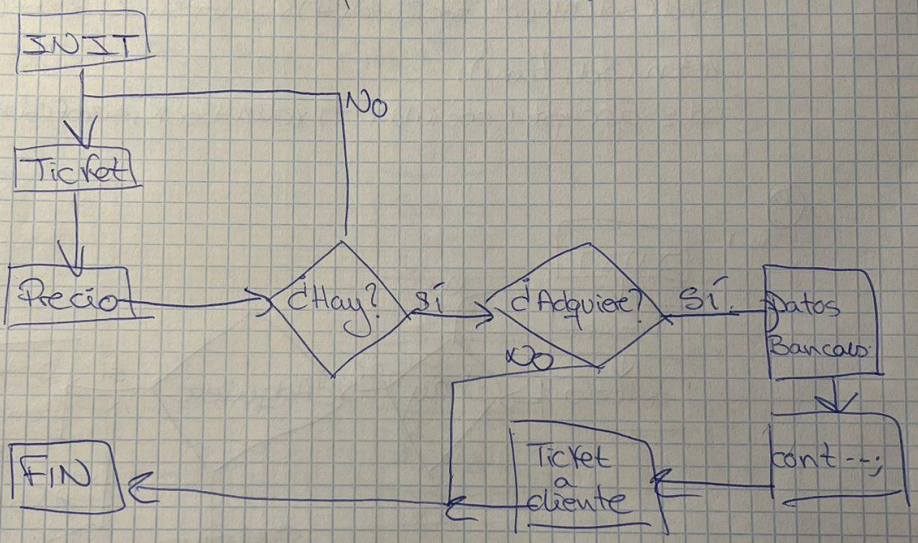
b) Info: {precio del ticket, *(disponibilidad), boolean (dispo) Id_ticket}.

c) Lo mas seguro sería usar semaforos para que no se pueda dar la posibilidad de que 2 usuarios compren el mismo ticket.

③
```
sem_t sem = sem_open("ticketSemaphore", O_CREAT, 1, 1);
sem_wait(sem);
char bd[];
while(validbd(bd)){
  printf("Información del banco no valida");
}
ticket->available = 0;
ticket->(*ptra)=0
sem_pos(sem);
```

④

Solo habría que cambiar que el ticket contenga una variable el nº posible de venta.



T.info: {Precio, *(disponibilidad), bool, Sol

Se sigue usando semáforos.

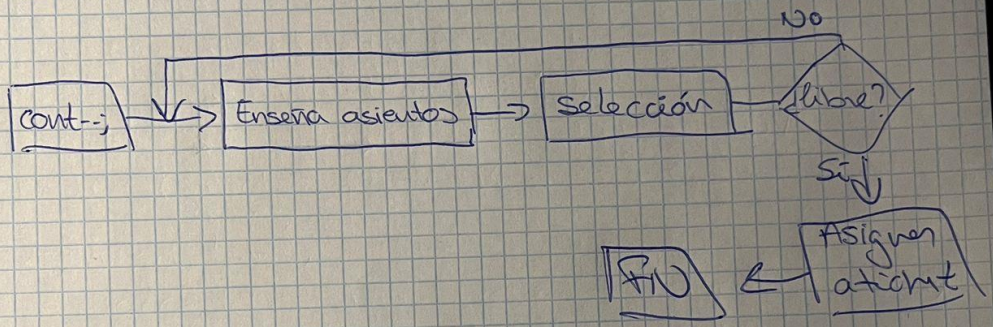
```
sem_t sem = sem_open("ticketSemaphore", O_CREAT, 1,1);
sem_wait(sem);
char bd[];
while(validbd(bd)){
   printf("Informacion bancaria no valida");
}
ticket-> available = ticket->*ptv --=1 )>0;
sem_pos(sem);
```

⑤

a) Ventajas:
- El cliente se asegura la compra del ticket
- la compra es mas rapida

Contra:
- Implementación mas complicada

b) El diagrama es el mismo, solo cambia en la sección crítica:



c) Al igual que se necesita espacio en memoria para los tickets, también haría falta para los asientos

Una estructura de datos para el ticket y otras para el cliente.