In [46]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [47]:
```
# Chargement du dataset
student_data = df = pd.read_csv("student-mat.csv",sep=';')
student_data.head()
```

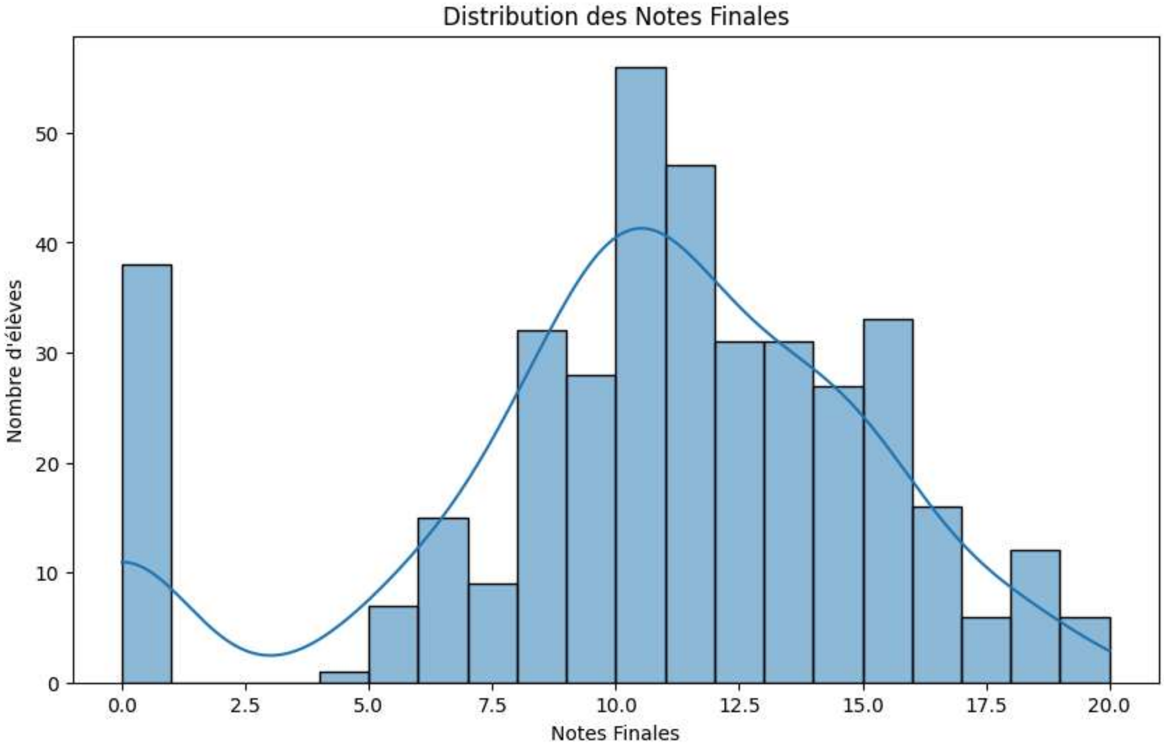
Out[47]:

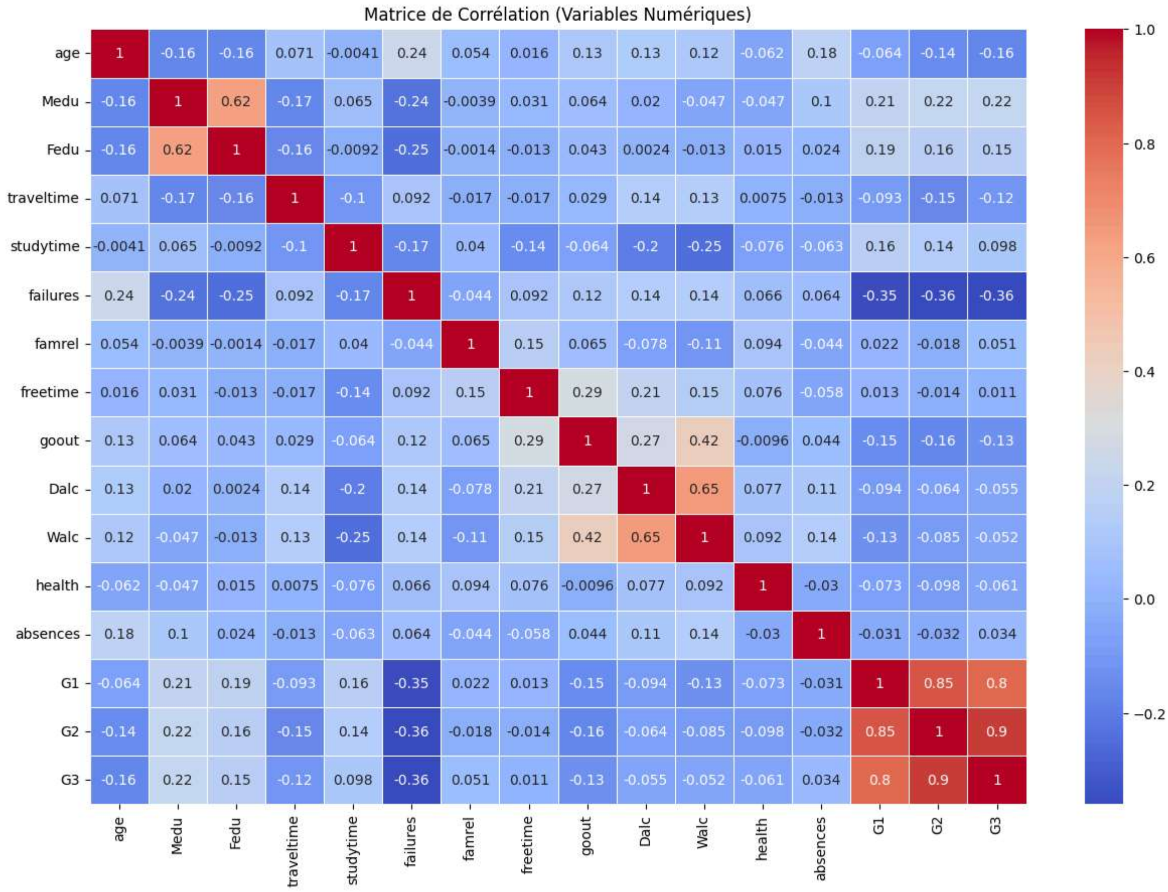
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	far
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	
3	GP	F	15	U	GT3	T	4	2	health	services	...	
4	GP	F	16	U	GT3	T	3	3	other	other	...	

5 rows × 33 columns

In [48]:
```
# Distribution des notes finales
plt.figure(figsize=(10, 6))
sns.histplot(student_data['G3'], bins=20, kde=True)
plt.title('Distribution des Notes Finales')
plt.xlabel('Notes Finales')
plt.ylabel('Nombre d\'élèves')
plt.show()

# Matrice de corrélation pour les variables numériques seulement
numerical_data = student_data.select_dtypes(include=['int64', 'float64'])
plt.figure(figsize=(15, 10))
sns.heatmap(numerical_data.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Matrice de Corrélation (Variables Numériques)')
plt.show()
```





In [49]:
```
# Encodage des variables catégorielles
label_encoder = LabelEncoder()
categorical_columns = student_data.select_dtypes(include=['object']).columns

for column in categorical_columns:
    student_data[column] = label_encoder.fit_transform(student_data[column])

# Normalisation des variables continues
scaler = StandardScaler()
numerical_columns = student_data.select_dtypes(include=['int64', 'float64']).col
```

```
student_data[numerical_columns] = scaler.fit_transform(student_data[numerical_co

# Séparation des caractéristiques (features) et de la cible (target)
X = student_data.drop(['G3'], axis=1)
y = student_data['G3']

# Division des données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [50]:

```
# Initialisation des modèles de régression
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Support Vector Regressor": SVR()
}

results = {}

# Entraînement et évaluation des modèles
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results[name] = {
        "Mean Absolute Error": mean_absolute_error(y_test, y_pred),
        "Mean Squared Error": mean_squared_error(y_test, y_pred),
        "R2 Score": r2_score(y_test, y_pred)
    }

results_df = pd.DataFrame(results).T
print(results_df)
```

```
                          Mean Absolute Error  Mean Squared Error  R2 Score
Linear Regression                    0.326834            0.240365  0.754578
Decision Tree                        0.298775            0.308347  0.685166
Random Forest                        0.246351            0.173132  0.823225
Support Vector Regressor             0.323536            0.251023  0.743696
```

In [51]:

```
# Entraînement du modèle
model = LinearRegression()
model.fit(X_train, y_train)

# Prédiction sur l'ensemble de test
y_pred = model.predict(X_test)

# Évaluation du modèle
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("R2 Score:", r2)
```

```
Mean Absolute Error: 0.3268340881789649
Mean Squared Error: 0.2403648826693127
R2 Score: 0.7545777855043496
```

In [ ]: